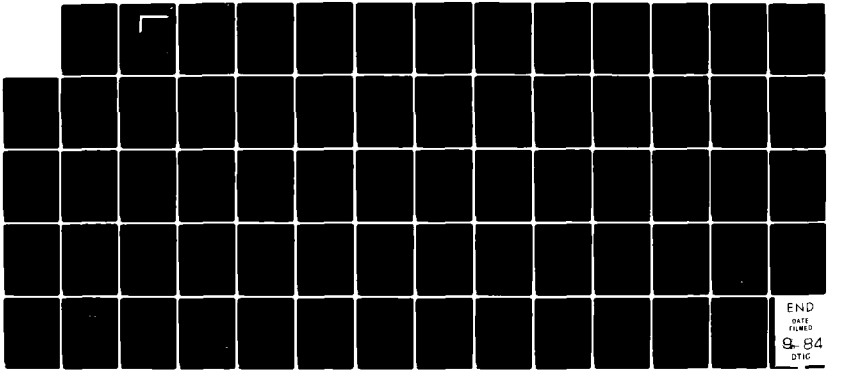
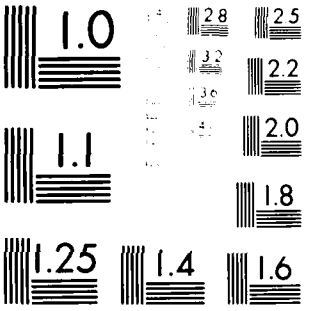
AD-A145 219 COMPUTATIONAL METHODS FOR COMPLEX FLOWFIELDS(U) 1/1
MASSACHUSETTS INST OF TECH CAMBRIDGE COMPUTATIONAL
FLUID DYNAMICS LAB E M MURMAN ET AL. 22 JUN 84
UNCLASSIFIED AFOSR-TR-84-0755 AFOSR-82-0136 F/G 12/1 NL



END
DATE FILMED
9-84
DTIC

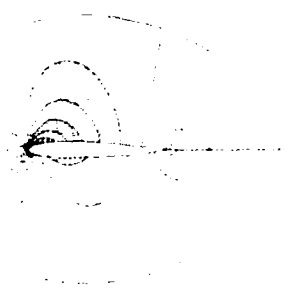


1984 ANNUAL REPORT

COMPUTATIONAL METHODS FOR COMPLEX FLOWFIELDS

Earll M. Murman
Judson R. Baron

Principal Investigators



COMPUTATIONAL FLUID DYNAMICS LABORATORY

Department of Aeronautics and Astronautics

Massachusetts Institute of Technology

Cambridge, Massachusetts 02139

(3) 2

1984 ANNUAL REPORT

COMPUTATIONAL METHODS FOR COMPLEX FLOWFIELDS

Earll M. Murman
Judson R. Baron
Principal Investigators

AFOSR Grant 82-0136
OSP 93729

June 22, 1984

Computational Fluid Dynamics Laboratory
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 0735	2. GOVT ACCESSION NO. AD-A145 219	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Computational Methods for Complex Flowfields		5. TYPE OF REPORT & PERIOD COVERED Annual Report June 1, 1983 - May 31, 1984 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Earll M. Murman Judson R. Baron		8. CONTRACT OR GRANT NUMBER(s) AFOSR-82-0136
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Aeronautics and Astronautics Massachusetts Institute of Technology Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2307/A1
11. CONTROLLING OFFICE NAME AND ADDRESS AFOSR/NA Bolling AFB DC 20332		12. REPORT DATE June 22, 1984 13. NUMBER OF PAGES 44
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* AFOSR/NA		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION. DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release. Distribution Unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Euler equations
Embedded grids
Adaptive grids
Airfoils
Computational Fluid Dynamics

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The development of solution algorithms for complex flowfields is the continuing objective of the research. Major focus is on use of coupled subdomains and descriptions which are either preselected or adapted to fit the physical events when necessary. The non-adaptive embedded mesh algorithm has completed airfoil solutions with an allowance for highly stretched meshes, alternate grids and reducing smoothing. A new algorithm is combining features from cell and nodal-centered methods to permit general embedded topology. Adaptive embedded mesh procedures have been extended to and carried out for

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. Abstract

two-dimensional Euler, subsonic, transonic and supersonic flows. An optimal distribution of local Courant numbers has been considered as a basis for accelerating the solution approach to a steady state.

TABLE OF CONTENTS

1. Introduction

A numerical simulation of a flowfield requires that the physical description and the coordinate grids both be consistent with the flow behavior. A number of methods now are available for different levels of physical approximation (such as Euler or Navier-Stokes) and for grid generation. However, only a small portion of a typical domain is subject to major disturbances. Therefore, control of descriptions and grids, their kind, location and extent, is of some practical interest for flow past arbitrary configurations.

An overall strategy is to subdivide the global domain into several local subdomains, each with its own equation and/or grid system. A suitable Computational Fluid Dynamics (CFD) algorithm would then have practical storage advantages and an increased speed of computation, but would require special stratagems to properly and efficiently couple the interacting subdomains. The simplest local equation system and coarsest grid are clearly most favorable but are subject to constraints related to resolving the flow details, preserving accuracy, and providing a correct modeling of different length scales that occur across the domain. Procedures which accomplish this for complex flowfields are the subject of this research.

2. Research Objectives and Tasks

The nature of CFD allows a flow solution to evolve in a discrete, spatial and temporal stepping fashion, which proves to be convenient for adjustment to the progress of either. The specific purpose of the research grant is to develop algorithms which control those adjustments throughout a field and at suitable or realistic intervals during the solution development, and on the basis of the actual local events and interactions that arise from a computation. This requires either a priori qualitative knowledge about the flowfield or procedures which can recognize scale differences and create or remove subdomains as the local need might demand or suggest.

It is therefore natural that separate tasks were undertaken to consider non-adaptive and adaptive embedded subdomains. Each task includes a global domain within which there are one or more embedded subdomains of the same topology and reduced grid scale. Non-adaptive implies preselection of realistic subregimes; adaptive implies solution-guided choices and updating of the grid configuration and the equations during the solution procedure.

The effort to date has carried this out using a basic algorithm (Ni's method, which is a conservative, finite volume, and multilevel approach) applied to channel and airfoil configurations, and two-dimensional Euler and Burgers equations. During the first year of the grant [1983 Annual Report, AFOSR-TR-83-0841] the emphasis was on necessary algorithm modifications for embedded regions, their interfaces, and their creation. During the past year the non-adaptive approach has been refined and an extended basic algorithm has been initiated to allow greater dimensional and topological freedom in applications. The adaptive approach has been extended to two-dimensional Euler system flows with and without discontinuities. In addition, a preliminary concept of a general accelerator for convergence to the steady state using explicit schemes has also been explored.

A brief summary of the recent work and implications appear in the three following sections. More detailed reports on each portion of the research are contained in the attached Appendices A, B and C.

3. Summary of Task I - Nonadaptive Embedded Subdomains

During the past twelve months our efforts have focussed in two directions. First, the work of Usab reported last year has been concluded with many important details being refined. Second, based upon the insight gained from our first experience with embedded mesh calculations, we have initiated the development of a more general algorithm for arbitrary 3-D grid topologies and vector or parallel computer applications. These two efforts are summarized below.

The work of Usab and Murman was reported in detail as Appendix A of last year's annual report. The essential findings therein regarding embedded mesh calculations have not changed. However, during the first six months of this year, additional work was completed so as to polish up the calculations and investigate several residual problem areas.

We had noticed that the basic Ni algorithm failed to converge on highly stretched meshes. The problem was traced to the use of a simple injection operator for the multiple grid step. That is, if the residual for the 2h grid was taken as the level h residual at the node point corresponding to the center of the 2h mesh cell, calculations failed to converge as the grid stretching increased. Various weighted averages of level h residuals at node points defining the 2h cell were tried, and a successful approach was found. It uses

the values of the level h residuals at the corner points of the 2h cell, but weighted by the appropriate second order terms in the Lax-Wendroff method. With this improvement, highly stretched grids could be used.

Our application of this was to study the effect of the far field vortex boundary condition for a lifting airfoil on a greater variety of grids than was previously possible. The following table from Usab's PhD thesis illustrates the importance of an accurate far field boundary condition (vortex and freestream) for a test case.

Variation of Force Coefficients with Location and Type of Far Field Boundary Condition (Actual values $C_L = 0.335$ and $C_D = 0.000$)

FAR FIELD RADIUS (CHORDS)	UNIFORM FREESTREAM BOUNDARY CONDITION		VORTEX FREESTREAM BOUNDARY CONDITION	
	C_L	C_D	C_L	C_D
5	0.2873	0.0030	0.3238	0.0019
10	0.3059	0.0022	0.3266	0.0016
20	0.3170	0.0016	0.3276	0.0013
30	0.3211	0.0013	0.3284	0.0011
50	0.3245	0.0010	0.3289	0.0009

A C-grid was applied to the airfoil problem so as to avoid the inherent O-mesh singularity at the sharp trailing edge. One result was that calculations could be obtained in half the number of iterations and with less added smoothing. In fact, it was possible to get a convergent calculation for a subcritical airfoil without adding any artificial viscosity. A sequence of five values of smoothing coefficients ranging from 0.0 to 0.8 were run and various parameters such as surface pressures and forces and surface total pressure losses were examined.

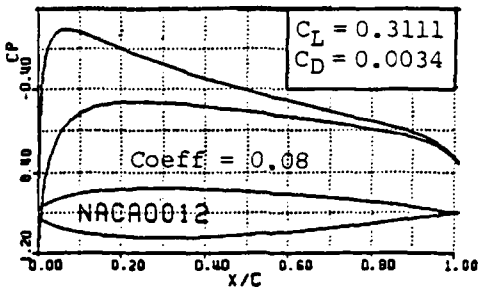
Figures 3-1 and 3-2 (from Usab's thesis) show how the lift, drag and total pressure loss calculations improve as the damping is reduced. The implications of this are very important for the achievement of accurate solutions for Euler equations modeling. Embedded mesh calculations also provide improved accuracy for a given smoothing model since the smoothing is always related to the mesh size.

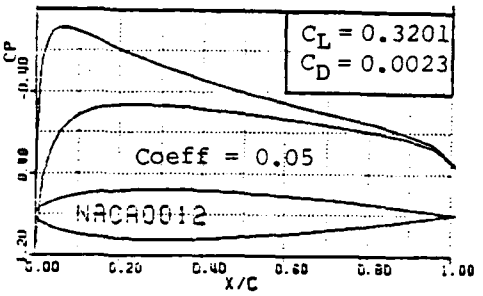
The findings of this phase of the work are summarized in detail in MIT-CFDL-TR-84-2 (also Usab's PhD thesis) given in the cumulative list of publications.

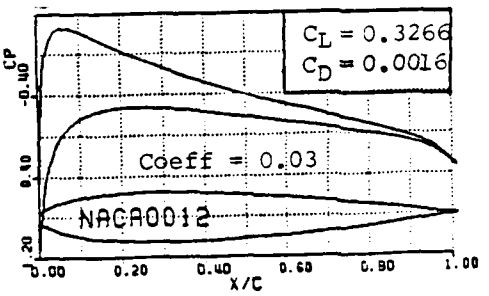
We are greatly encouraged by the potential that embedded mesh approaches offer. Yet the work of Usab is not directly extendable to (a) three-dimensions, (b) embedded grids of non-similar topology, and (c) vector or parallel processing. The basic reason for these limitations is that the pointer structure used in that work was not intended to be completely general, and Ni's algorithm is not directly extendable to grids which are non-mappable to a Cartesian space. With these requirements in mind, we set out to develop a second generation method.

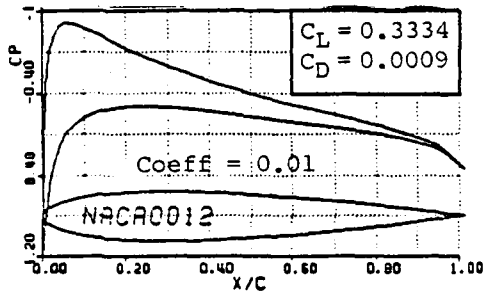
After considering both cell-centered methods (such as Jameson) or nodal methods (such as Ni), we decided that nodal methods offer higher accuracy with less added complexity. However, the multistage algorithms of Jameson offer more flexible time integration methods than the Lax-Wendroff approach and they should not be limited to cell-centered schemes. Our new method thus takes what we feel are the best attributes of both approaches. Some preliminary results have been obtained for a simple test problem. They are encouraging but indicated that we need to concentrate now on the damping.

A great deal of thought has been given to the data base management and pointer systems. At present we have a general approach which as coded









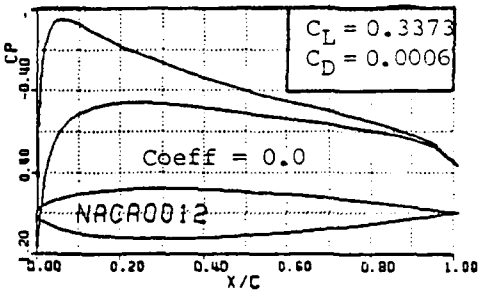


Figure 3-1. (from Fig. 3-22 of Usab's thesis) Comparison of surface pressure coefficient for various smoothing coefficient values (0.0 to 0.08) for C-mesh. M = 0.630, α = 2.0°

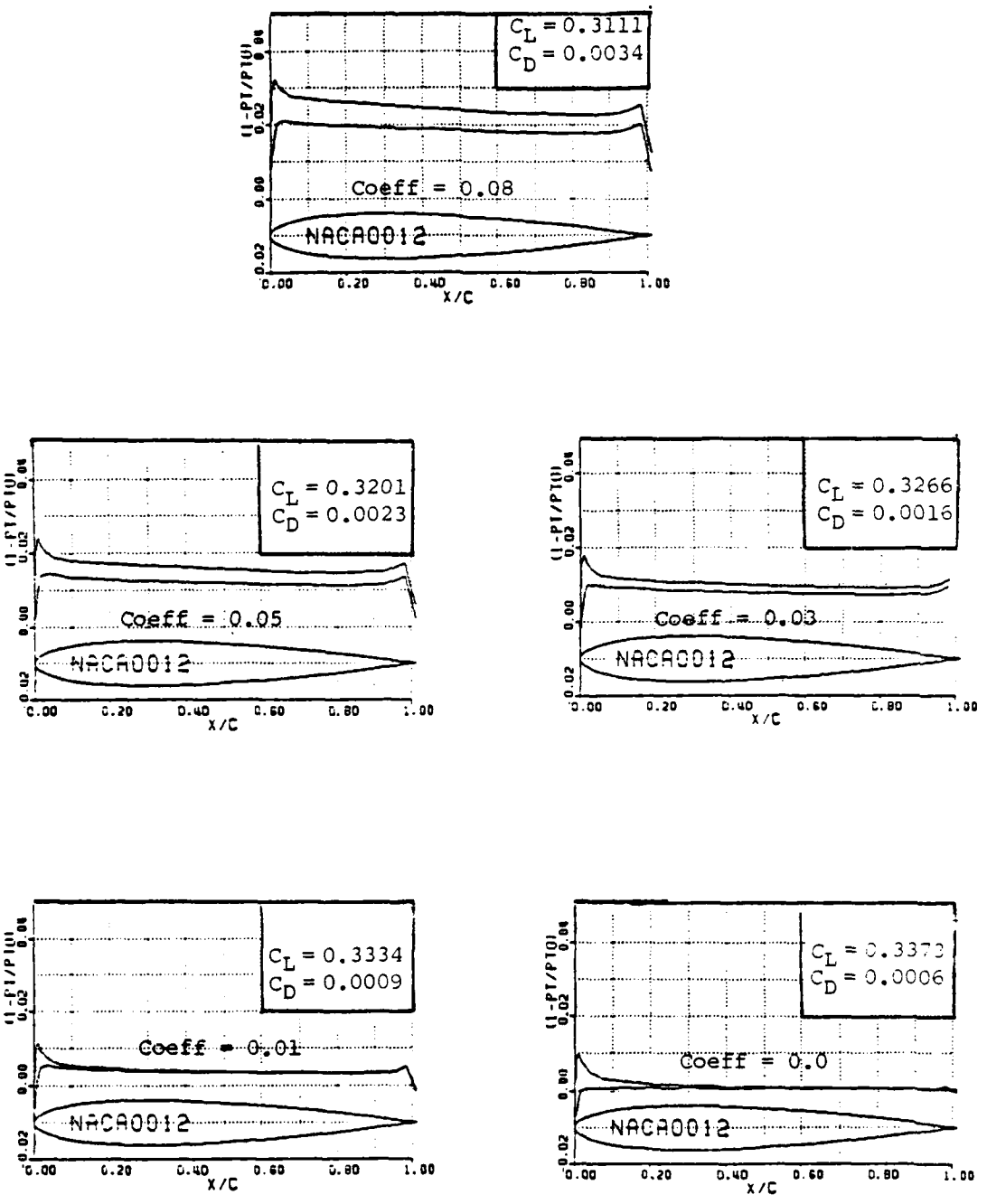


Figure 3-2. (from Fig. 3-23 of Usab's thesis) Comparison of surface total pressure loss for various smoothing coefficient values (0.0 to 0.08) for C-mesh. M = 0.630, α = 2.0°

will be suitable both for 2 or 3 dimensions, and for grids which need not be mappable to a Cartesian space. In addition we believe that it is adaptable to rather arbitrary machine architectures. The details of this are given in Appendix A. However, our experience with this algorithm is very limited at this point. During the next year we will be developing its capabilities.

4. Summary of Task II - Adaptive Embedded Subdomains

The adaptive concept developed during the first grant year established the utility of embedding finer grid structures and alternate descriptions within the framework of a global algorithm and only if and when required. The first part of Appendix B is a description of that work as presented at an AIAA meeting. The essential findings were based upon model problems, purposely kept simple in the one dimensional (inviscid streamtube) and single variable (Burgers equation) sense. However, the recognition of important features, the manipulation of grid structures, and the change of governing equations all were included and shown to be effective. A measure of this is a factor of 3 computation time advantage that resulted for the adaptive one dimensional problem.

During the last year the concept has been extended to two-dimensional Euler systems and test cases have been completed for flow past circular arc sections in a channel. Subsonic, transonic and supersonic flows were considered in order to include discontinuities of different strengths, orientations, and multiplicity. A discussion of the two-dimensional considerations and some results are contained in the second part of Appendix B and are summarized below.

- A single parameter basis provides an adequate criteria for feature recognition in a two-dimensional Euler system.
- Threshhold level is a factor in, but not critical to, the recognition of multiple features or their interacting regions.
- Irregular embedded grid patterns present no problem for curved features.
- Pointer systems for collapsed two-dimensional systems require special knowledge of adjacent cell/node locations for final adjustment of floating, collapsed features.

° Variable smoothing coefficients have been introduced to adapt the smoothing effect to the need.

° Interpolation and floating schemes for fitted, curved shock discontinuities have been developed but not yet activated in operational, two-dimensional adaptive codes.

The cases evaluated to date indicate appreciable computing time benefits with only grid adaptation being active. Factors of 10. are mentioned in Appendix B for a transonic flowfield. The present objective is to include the shock fitting procedure within the adaptation sequencing, allow for either kind of adaptation in the likely event of multiple features, and consider "overlapping" (i.e. interacting) features and their appropriate edges. The intended application is to an airfoil at angle of attack.

5. Summary of Convergence Acceleration Concept

The research described above assumes that interest is centered on the steady state solution. The use of a multilevel acceleration technique precludes developing a time accurate iteration history. Similar accelerators to hasten convergence have been suggested for explicit finite difference schemes. Appendix C describes some preliminary work that suggests a possible substantial improvement over the well known approach which advances the solution using the local maximum Courant numbers that maintain stability. Consideration was given a model partial differential equation (first order, nonlinear) and a rational basis was used to define an optimal distribution of local Courant numbers. Numerical comparisons showed convergence to the steady state to be achieved with an order of magnitude reduction in the number of pseudo-temporal iterations. The method is now being extended to and explored for a multidimensional system and various difference schemes. This effort was carried out by Professor Saul S. Abarbanel with the collaboration of D. Gottlieb, whose interest overlaps from his participation in another OSR grant.

6. Cumulative List of Publications

[1] Usab, W.J. and Murman, E.M., "Embedded Mesh Solutions of the Euler Equation Using A Multiple Grid Method." AIAA Paper 83-1946 CP, July 1983. Also to appear in RECENT ADVANCES IN NUMERICAL METHODS IN FLUIDS, Vol. 3, Editor W.G.Habashi, Pineridge Press.

[2] Usab, W.J., "Embedded Mesh Solutions of the Euler Equations Using A Multiple-Grid Method." MIT PhD Thesis, Jan. 1984. Also MIT CFDL-TR-84-2, May 1984.

[3] Dannenhoffer, J.F. and Baron, J.R., "Adaptive Solution Procedure for Steady State Solution of Hyperbolic Equations." AIAA Paper 84-005. Also submitted to AIAA Journal.

[4] Dannenhoffer, J.F. and Baron, J.R., "Adaptation of Equations and Grids for the 2-D Euler Equations." Submitted to AIAA Aerospace Sciences Meeting, January 1985.

7. Professional Personnel Associated with Research Effort

Principal Investigators

Earll M. Murman, Professor
Judson R. Baron, Professor

Others

Saul S. Abarbanel, Senior Lecturer,MIT; Professor, Tel-Aviv University
William J. Usab, Jr., PhD candidate (degree awarded Feb. 1984)
John F. Dannenhoffer III, PhD candidate
Gregory L. Larson, PhD candidate

8. Interactions

1. Papers, seminars, presentations

   . AIAA Paper 83-1946CP presented at AIAA 6th Computational Fluid Dynamics Conference, Danvers, Mass., July 1983.
   . AIAA Paper 84-005 presented at AIAA 22nd Aerospace Sciences Meeting, Reno, Nevada, January 1984.
   . Seminar on Adaptive Embedded Subdomain results in Paper 84-0005 presented by Baron at University of Maryland, College Park, Md., February 1984.

2. Interactions with DOD Laboratories

   . Seminar on Nonadaptive and Adaptive Embedded Subdomains presented by Murman and Baron at AFWAL, Dayton, Ohio, March 1984.
   . Conferences with Dr. Wilbur Hankey, FDL.

9. New Discoveries, Inventions, etc.

None

APPENDIX A

NONADAPTIVE EMBEDDED SUBDOMAINS

1. MOTIVATION FOR NEW ALGORITHM

Many physical systems can be modeled by an equation of the form

$$\bar{U}_t = -\bar{\nabla} \cdot \bar{\bar{F}} \tag{1}$$

In particular, if the state and flux terms are

$$\bar{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \quad \bar{\bar{F}} = \begin{pmatrix} \rho u & \rho v & \rho w \\ \rho u^2+p & \rho uv & \rho uw \\ \rho uv & \rho v^2+p & \rho vw \\ \rho uw & \rho vw & \rho w^2+p \\ \rho uH & \rho vH & \rho wH \end{pmatrix} \tag{2}$$

then Eq(1) is a statement of the Euler equations for three-dimensional inviscid fluid flow. Equations of this type can be conveniently discretized and solved by means of a finite volume method, in which the physical space is divided into nonoverlapping, or disjoint, finite cells, on each of which the right hand side of Eq(1) is required to vanish at steady state. The equation is integrated over each cell, and the right hand side is transformed from a volume integral into a surface integral by Green's theorem:

$$\int_{\text{volume}} \nabla \cdot \bar{\bar{F}} dV = \int_{\text{boundary}} \bar{\bar{F}} \cdot d\bar{n} = \sum_{\text{faces}} \bar{\bar{F}} \cdot \bar{n} \tag{3}$$

in which the summation is over the faces that bound the cell.

Within the general class of finite volume methods there are two methods which are currently popular, each representing a different subclass. Jameson's method is the most popular version of a cell-based scheme, and Ni's method is a well known example of a node-based scheme.

1a. Jameson's Method

In cell-based schemes, the state vector is associated with the center of the cell (Figure A1). This causes some minor difficulty in the evaluation of the right hand side. In order to evaluate the flux F at a face, the state U at that face must be known first. The usual procedure is to take a simple average of the states at the cells on either side. This seems reasonable enough, and is second order accurate in the cell size if the mesh is uniform, but it is only accurate to first order in the cell size if the mesh is skewed or nonuniform. The left hand side can be solved to second order as

$$\int \bar{U}_t dV = \bar{U}_t V \tag{4}$$

where V is the volume of the cell. Thus, to an order of accuracy in space limited by the accuracy of the flux balance,

$$\bar{U}_t(\bar{U}) = - \frac{\Sigma \bar{\bar{F}}(\bar{U}) \cdot \bar{n}}{V} \tag{5}$$

It remains only to solve the resulting coupled ordinary differential equations, which Jameson does with the following multistage algorithm.

$$\begin{aligned} U_o &= U^n \\ U_j &= U^n + \alpha_j (\Delta t) U_t(U_{j-1}) \quad [j = 1, \ldots, k] \\ U^{n+1} &= U_k \end{aligned} \tag{6}$$

1b. Ni's Method

Ni's method is a nodal scheme (Figure A2). The state vectors are associated with the nodes which define the cell corners. With the state vectors known at the corners of the cells, the fluxes can be evaluated to second order accuracy by the trapezoidal rule, regardless of the nonuniformity or skewness of the mesh. However, the left hand side of the governing equation can no longer be solved explicitly for U_t. Ni therefore uses a distribution formula which is based on a Lax-Wendroff iteration. That is, from Eq(1),

$$\bar{U}_{tt} = -(\bar{\nabla} \cdot \bar{\bar{F}})_t = - \left(\bar{\nabla} \cdot \frac{\partial \bar{\bar{F}}}{\partial \bar{U}} \right) U_t = \bar{\nabla} \cdot \left[\frac{\partial \bar{\bar{F}}}{\partial \bar{U}} (\bar{\nabla} \cdot \bar{\bar{F}}) \right] \tag{7}$$

Then, the time integration is achieved by using an explicit three-term Taylor series for the next time level in terms of the current one.

$$\begin{aligned} \bar{U}^{n+1} &= \bar{U}^n + \Delta t \bar{U}_t + \frac{(\Delta t)^2}{2} \bar{U}_{tt} \\ &= \bar{U}^n - \bar{\nabla} \cdot \bar{\bar{F}} + \bar{\nabla} \cdot \left[\frac{\partial \bar{\bar{F}}}{\partial \bar{U}} (\bar{\nabla} \cdot \bar{\bar{F}}) \right] \end{aligned} \tag{8}$$

In the evaluation of this expression, the term $\bar{\nabla} \cdot \bar{\bar{F}}$ is evaluated conservatively, to second order. The second derivative, however, is evaluated by taking a first derivative of the quantity in square brackets, which is second order accurate only if the mesh is smoothly varying. Therefore, Ni's method might have convergence problems on highly stretched meshes due to the inaccuracy of the second derivative calculations.

Realistic problems are expected to have meshes that are highly distorted. Even for a problem as simple as an airfoil, a C-mesh is frequently used to avoid the distortion involved in wrapping an O-mesh around the trailing edge, in spite of the lower density at the trailing edge that a C-mesh produces. Therefore, a preferable method would make use of only simple formulas to get second order accuracy, no matter how badly distorted or irregular the grid. Noting that neither Ni's spatial nor Jameson's temporal discretizations would cause any problems in that attempt, it was decided to try a formulation which combined the best of both methods. In the process, care was to be taken to make sure that all calculations were carried out in a way that could easily be assigned to parallel processors.

2. MOTIVATION FOR POINTER SYSTEM

2a. Complicated Geometry

Until recently, most fluid flow calculations were carried out on grids that mapped the physical domain onto a rectangle in computational space. That is, the physical space would be represented by a two or three dimensional array of points. When the computations at a point (I,J) required information from an adjacent point, to the right, for example, that information was found at the position (I+1,J) in the array, as shown in Figures A1 and A2. This method works well only if a mapping from physical to computational space can be found which concentrates points where extra resolution is needed. However, that may prove to be difficult or impossible with some of the physical geometries of interest. An alternative is the use of pointers to keep track of connectivity. In the calculation of some quantity at the point I, pointers associated with I point to neighboring values required by the algorithm.

2b. Desired Characteristics

Many different structures could be imagined for a pointer system, with the choice depending on precisely what information is needed in connection with the calculations. Usab used a pointer system in which each cell pointed to exactly nine nodes, some of which could be nonexistant (Figure A3). The first four nodes pointed to from each cell were the corner nodes, which always exist. The remaining five were the nodes at the middle of each face and at the center of the cell, which might or might not exist, depending on whether or not the cell or any of its immediate neighbors is subdivided. This particular choice of pointers works only if all cells are four sided and all mesh refinements or multigrid mesh inclusions are binary. This may seem like an innocuous requirement, but one could easily imagine a grid composed of a Cartesian far field grid with embedded body fitted meshes wrapped around the physical features of interest. At either the interfaces between the embedded meshes or between an embedded mesh and the far field, it might be very difficult to guarantee four sides to every cell (Figure A4). In three dimensions it would almost certainly be impossible to guarantee each cell its six faces and eight nodes. It was decided, therefore, to use a pointer structure which made a minimal number of assumptions about the structure of the computational grid, while still making available enough connectivity information to the program so that it can perform its computations without excessive searching for neighboring data.

The computational grid for any conservative finite volume computation can be viewed as consisting of disjoint cells, faces and nodes (Figure A5). Since there are parameters associated with each of the three constituents of the mesh, there are data substructures in the pointer system for cells, for faces and for nodes. Each cell is physically defined by the faces that bound it, so the data structure for each cell includes the number of faces defining that cell and pointers to those faces. Each face is defined by the nodes that bound it, so the data structure for each face includes the number of nodes and pointers to those nodes. Thus, a grid is defined in a topological sense by pointers from cells to faces and from faces to nodes. Further geometric information that may be needed includes the volume of the cells, the directed normal area of the faces, and the (x,y) or (x,y,z) position of the nodes.

Although this is sufficient to define the mesh, both topologically and geometrically, pointers also have been included from nodes back to cells. Any adaptation or shock fitting would be greatly simplified if the pointers form a ring, so that a cell can interrogate faces, then nodes, to detect its own neighbors. Also, distributions to nodes in the proposed algorithm will be greatly facilitated by pointers from nodes to cells.

Note that this pointer system is specifically designed for the proposed algorithm with Ni's spatial discretization and Jameson's time operator. A code using Jameson's method would not require the data structure for nodes, since there are no fundamental quantities associated with nodes. In that case, it would make sense to have pointers from cells to faces, and from faces to cells.

Since the eventual intent is to produce working three-dimensional codes, care is being taken during the development of both the pointer system and the algorithm to avoid any reference, either in the analysis or in the coding, to dimensionality. In doing a flux integral over a cell, for instance, the loop is not over the four or six faces of the cell, but over the number of faces specified for that cell by the mesh generator. For the inner product of a flux vector with a normal area vector for a face, a loop appears as

```
      DO 100 L=1,NEQS
      FLOW(M)=0
      DO 100 M=1,NDIMS
      FLOW(M)=FLOW(M)+FLUX(L,M)*NORMAL(M)
  100 CONTINUE
```

where NDIMS and NEQS are compile time parameters for the number of components in the spatial and state vectors respectively. Although one might argue that this is much less efficient than writing the expressions out, it proves to be much more efficient in terms of programmer time; a good optimizing compiler expands the small loops automatically in any case, so there is no actual loss of performance. Thus, if the code proves its value in a 2-D Euler flow, very few lines of code need to be changed to upgrade to either 3-D (increasing the spatial and state vector length) or to MHD (which would only increase the state vector length).

3. BASIC ALGORITHM

The algorithm consists of a flux balance over the cell surface to determine the time derivative of the state vector, followed by a distribution of changes from cells to nodes, all embedded in a multistage time integration. Separate explicit smoothing and boundary enforcement are necessary.

3a. Flux Balance

The flux balance proceeds as a loop over the faces. On each face, the state is calculated as a weighted average of the states at the defining nodes. In two dimensions, the weights are always equal, and could be written into the program as 0.5. However, in three dimensions, there could be various numbers of nodes defining different faces, and there is no reason to assume a priori that they should be equally weighted. Therefore, the algorithm uses a weight

for each node that is calculated by the mesh generator. First a state, then the flux vector for that state, and then its inner product with the normal area vector for that face are calculated, and that quantity is saved. Next, looping through the cells, each cell collects the integrated flux from its defining faces. At this point, each cell has its divergence of the flux vector, integrated over the cell volume. This quantity is multiplied by the allowable time step for the cell and divided by the volume to give the change in state that should hold for the cell.

3b. Distribution

There are a number of possible ways of interpolating the changes from the cells to the nodes. Among those, only those that reduce to a simple averaging on a regular mesh can be considered. The one which is currently under investigation is to weight by the subtended angle (Figure A6). This has been chosen largely because it is easy to calculate using the same information that is used by the smoothing operator.

3c. Time Integration

The time integration is the same as in Jameson's scheme. The flux integral and distribution steps, taken together, define the change in the state vector, ΔU, as a function of the state vector U. The several intermediate stages are defined as

$$U(0) = U_n$$
$$U(j) = U_n + \alpha(j) * \Delta U[U(j-1)] \qquad [j = 1,2,\ldots,k]$$
$$U_{n+1} = U_n + \Delta U[U(k)]$$

Note that this equation set is nearly the same as Eq(6). The difference lies in the fact that in the extended method there is no time derivative of U at the nodes. Therefore, the quantity ΔU replaces $(\Delta t)(dU/dt)$.

The optimal number of stages and the coefficients α_j have yet to be determined. It is planned to use parameter optimization to find the optimal coefficients for each number of stages, and to compare the reduction in residual per cpu second for each number of stages to find the parameters.

3d. Boundaries

Two additional steps are necessary at boundaries. At an open boundary, or fictitious boundary between the discretized portion of the flow field and the far field, an approach is used based on Godunov's method. During the flux integral phase, a one-dimensional simple wave problem is solved after the state at the face is found by averaging from the nodes, and before the calculation of the flux vector. First, the velocities are projected into normal and tangential components. Second, on the basis of the known states inside and outside, the velocity of the resulting contact discontinuity is found. (For wall boundaries, this is known a priori to be zero.) With the velocity known, the density and total volumetric energy can be solved for on either side of the contact discontinuity. If that velocity is positive, i.e. fluid flowing out through the

boundary, then the new state vector is the one calculated on the inside of the discontinuity. Otherwise it is the state immediately outside. The applicable tangential velocity is also the one from the appropriate side of the discontinuity.

The other measure taken to ensure satisfaction of the boundary conditions is a simple wave solution applied to the nodes after each iteration. The normal direction is taken to be the sum of the normals of the faces on each side of the current node. Other than that, the procedure is the same as that applied at the faces.

3e. Smoothing

Some form of smoothing is necessary, since the algorithm described above is completely transparent to sawtooth waves. An optimum smoothing procedure has not yet been determined. One form has been found which damps out the sawtooth waves, but has excessive damping. The kind of smoothing operator being considered is a polynomial in a kind of second difference operator, which is calculated as follows. The values are averaged from the nodes to the faces, and then to the cells. Then the difference is calculated by summing the differences between the values at the cell and at the node, multiplied by the angle (solid angle in 3-D) subtended by the cell from the node.

$$D^2(U) = \sum \left(U_{cell} - U_{node}\right) \theta_{cell} \tag{9}$$

The operator in use now is of the form

$$U_{smooth} = \left(1 + \nu D^2\right)^2 U_{rough} \tag{10}$$

It is hoped that an operator can be implemented which would provide more fourth difference smoothing with less second difference smoothing, such as

$$U_{smooth} = \left(1 - \nu\left(D^2\right)^2\right) U_{rough} \tag{11}$$

3f. Acceleration

No attempt has been made as yet to implement a multigrid accelerator. However, there does not seem to be any reason to expect it not to work as well here as it has with Ni's and Jameson's methods.

4. RESULTS

Solutions to the McCartin (Ni's) bump problem have been generated which are consistent with an interpretation that the imposed smoothing is overly dissipative. The two calculations that were performed on a 65 x 17 mesh yielded spectral radii of 0.9913 and 0.9918. This is without adjustment of the free parameters, i.e. the α coefficients for each stage in the time integration, and the CFL number. The spectral radius would probably be

larger, and the convergence therefore slower, if the smoothing were not too large. Both cases were run using two stages of equal weight.

5. VECTORIZATION/PARALLEL PROCESSING

The entire algorithm falls logically into separate phases, on each of which a large number of calculations could be performed in parallel. For instance, the calculation of the flux vector, and its inner product with the normal vector to a face, are carried out independently for each face. A large number of processors could simultaneously consider different faces. Similarly, the calculation of the changes at each cell or the updating of the state vector at the nodes could be accomplished in parallel.

6. SUMMARY/CONCLUSIONS

A new computational method for solving conservative partial differential equations is being developed. The method has been formulated in such a way that it should be completely mesh independent, in the sense that it would operate on any finite volume mesh that can be generated, including highly stretched, skewed, or irregular meshes, and still retain second order accuracy in cell size. Preliminary results have been obtained solving the Euler equations on the McCartin bump problem.

In the near future, work will concentrate on the following areas:

1. Improving the smoothing operator
2. Implementing a multigrid accelerator
3. Determining optimal values for the free parameters
4. Developing grid generators to take advantage of the flexibility of the algorithm
5. Solving realistic one and two body airfoil problems
6. Solving a simple three dimensional problem

It is felt that with the correction of the smoothing operator, the addition of multigrid acceleration, and the implementation of well thought out grid generators, the algorithm will provide a relatively efficient solver for complex two and three dimensional inviscid flow fields. The extension to viscous flow appears to be straightforward.

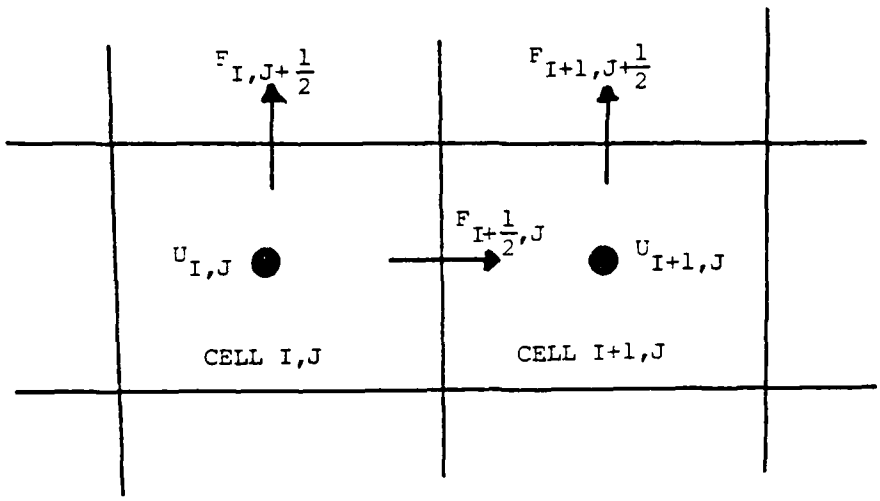


Figure A1. Cell based scheme (2-D)

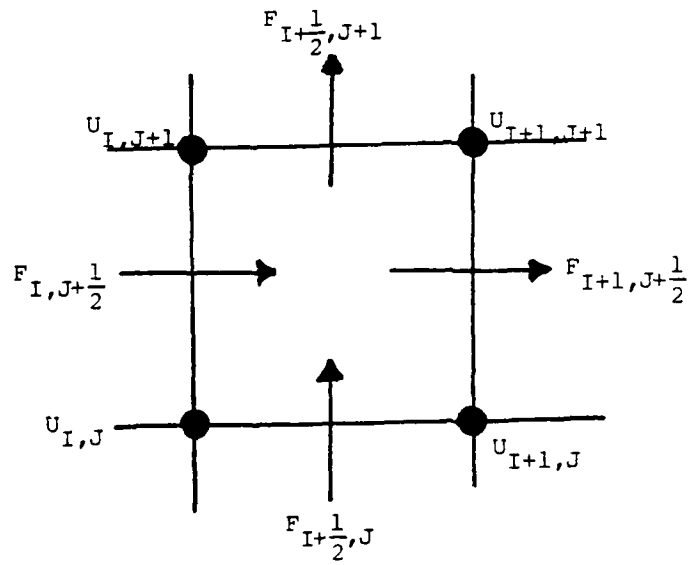
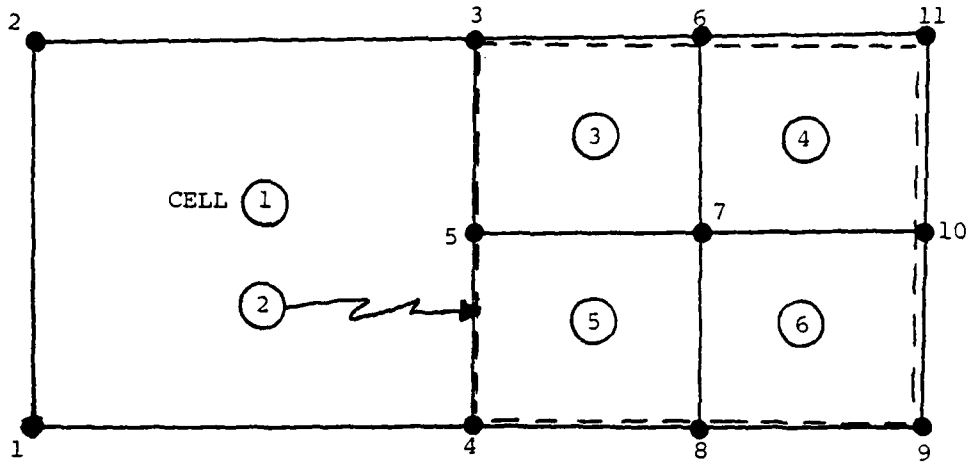


Figure A2. Node based scheme (2-D)



CELL	NODES
1	1, 2, 3, 4
2	4, 3, 11, 9, 7, 5, 6, 10, 8
3	5, 3, 6, 7
4	7, 6, 11, 10
5	4, 5, 7, 8
6	8, 7, 10, 9

Figure A3. Cells and defining nodes

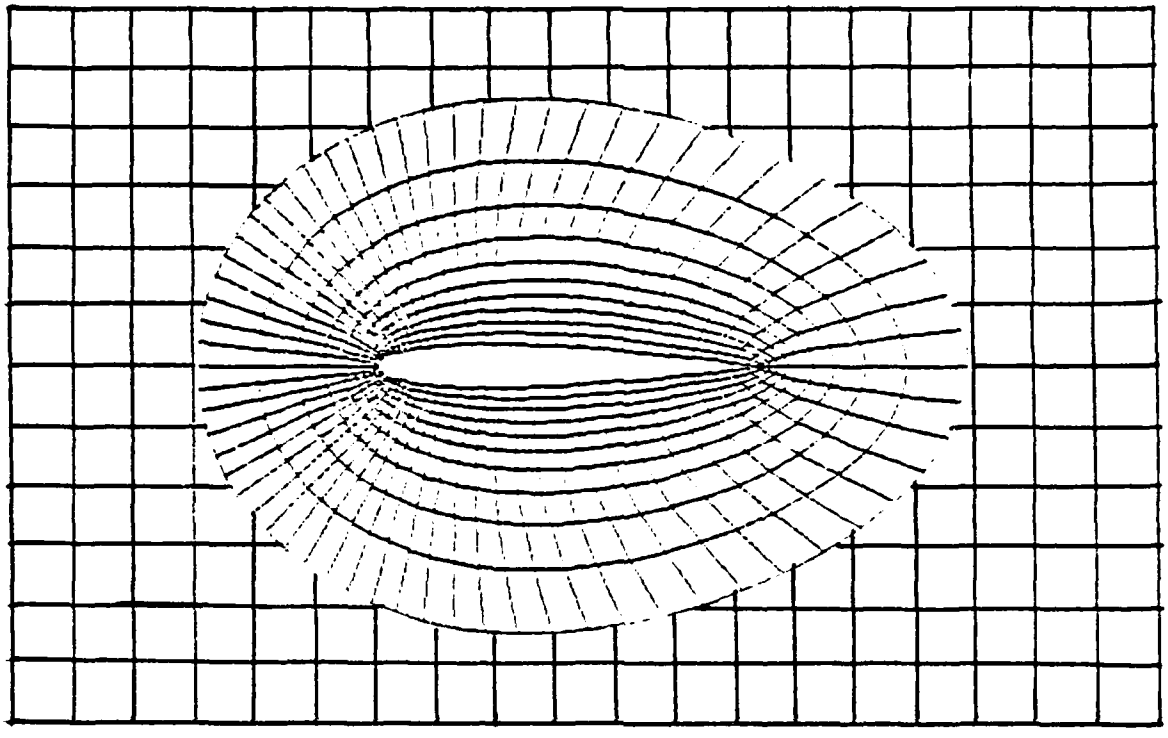


Figure A4. Topologically mismatched embedded grids

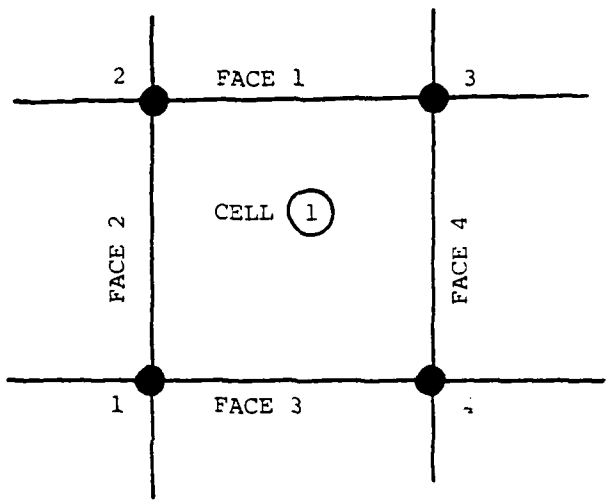
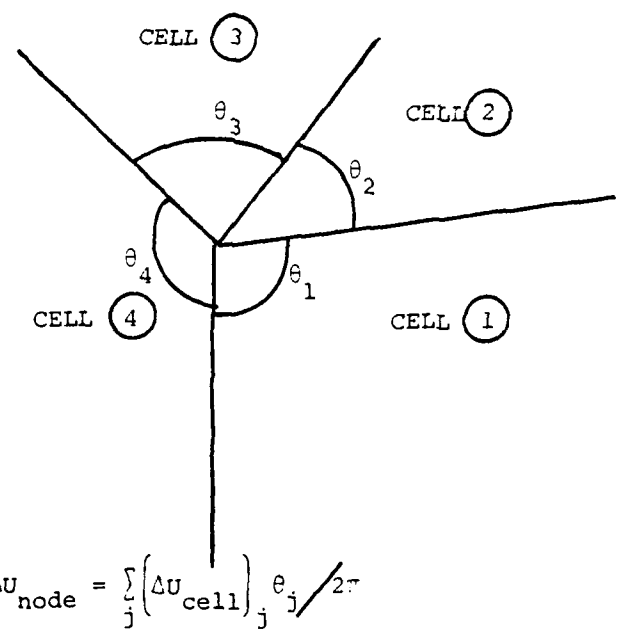


Figure A5. Faces and nodes



$$\Delta U_{node} = \sum_j \left(\Delta U_{cell}\right)_j \theta_j / 2\pi$$

Figure A6. Subtended angle weighting

AIAA'84

AIAA-84-0005
Adaptive Procedure for Steady State Solution of Hyperbolic Equations

J.F. Dannenhoffer III and J.R. Baron,
Massachusetts Institute of Technology,
Cambridge, MA

AIAA 22nd Aerospace Sciences Meeting

January 9-12, 1984/Reno, Nevada

ADAPTIVE PROCEDURE FOR STEADY STATE SOLUTION OF HYPERBOLIC EQUATIONS

AIAA-84-0005

John F. Dannenhoffer, III *
Judson R. Baron **

Computational Fluid Dynamics Laboratory
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

An algorithm to determine the steady state solution of a set of hyperbolic partial differential equations using grid- and/or equation-adaptation is presented. The evolving solution is periodically examined for isolated features and the adaptation strategy most appropriate to each is applied. Simple detection and pattern recognition procedures are used to locate such features. Grid adaptation is accomplished by using an embedded mesh procedure on an irregularly shaped embedded domain, with a multiple-grid accelerator used to couple the global and embedded regions. Equation adaptation is accomplished by altering the basic integration scheme in cells through which "collapsible" features pass. A flexible data structure is employed to make switching between the two adaptation schemes relatively simple. A new program has been written for general hyperbolic systems in two dimensions. Specific computed examples include the one-dimensional Euler equations with both adaptive equations (a new floating shock fitting procedure) and adaptive gridding. The Burgers equation is used to illustrate the adapted grid solution on an irregular embedded region in two dimensions. Significant time savings and accuracy improvements are shown to be achievable with the present method.

INTRODUCTION

In general, the steady state solution of a system of hyperbolic, partial differential equations can be computed by a discrete approximation to the governing equations. The discreteness of the approximate equations causes truncation errors which are typically related to both the local computational grid size and the local solution behavior. For purposes of computational efficiency, the mesh on which the discrete approximations are made is generally coarse.

For problems of practical interest, the truncation errors over a large portion of the domain are frequently small due to smoothness of the solution. However, there can be features within the domain where the solution is not smooth; shock waves, boundary layers, and wakes are examples. To minimize the truncation errors which are generally much larger near the feature than the average over the whole domain, one can either introduce a finer computational grid or alternatively provide a more precise accounting of the local physics which is responsible for the local non-smoothness.

Unfortunately, one does not know a priori where the features occur within the domain. Adaptive solution algorithms is a generic designation for numerical methods which sense unique physical behavior (features) in the flowfield being computed, and subsequently change the governing equations and/or computational grid to adequately describe those features. Adaptive solution algorithms which change the governing equations are known as adaptive equation techniques; those which alter the computational mesh are known as adaptive grid techniques.

Initially, consider only adaptive grid procedures. These schemes make use of the same descriptive equations as before adaptation, but with the grid spacing adjusted so that the integration algorithm properly captures the local physics. For example, an adaptive grid algorithm captures boundary layer gradients by decreasing the local mesh spacing at the wall.

Within the basic framework of adaptive grid algorithms, there are currently two major approaches. The first (and currently most popular) can be called grid point redistribution, and the second can be called embedded grids.

Many different grid point redistribution schemes have appeared recently in the literature [1-13]; most are discussed in the survey article by Thompson [14]. In these schemes, a fixed number of grid points are redistributed throughout the flow field. The nodes are moved about such that grid points are concentrated in regions of high gradients or errors (most likely regions near features). For one-dimensional cases, this results in a solution which may be viewed as either uniformly good or alternatively uniformly bad, since in regions with small errors initially, the grid point are "moved out", resulting in

* Research Assistant, Member AIAA
** Professor, Associate Fellow AIAA

larger errors there. In cases with multiple features, the time development of the features can have a large influence on the "split" of mesh points between the features and hence their resolution. Further complications are added in two-dimensional cases due to topological limitations; for example, grid lines concentrated near a shock at an airfoil surface must also propagate away from the airfoil, resulting in excessive resolution in the farfield.

One way that the redistribution is accomplished is by treating each node as a body, assigning an attraction parameter to each, and then integrating the resulting n-body problem forward in time [1-4]. Special care must be taken in such techniques, since unfortunately grids may result which cross and/or are excessively skewed [1]. To circumvent these problems, some have resorted to solving partial differential equations [11] while others have posed the grid generation as a variational problem with direct control over grid concentration, smoothness, and skew [12]. In any event, the result of grid point redistribution is a grid with varying grid spacing and skew. It has been shown that both of these can cause significant errors in the computed solution [15]. Also, care must be exercised to ensure that the moving grid and the flow solution do not couple, yielding the unwanted oscillations reported by some [7]. In addition, displacing grid points implies that the previously calculated values at previous grid point locations are no longer correct.

Despite these drawbacks, this method is very popular since the logic needed to implement the algorithm is relatively straightforward. The same logic can be used for all cases, independent of the features which are (or are not) present.

The other adaptive grid algorithm is known as embedded grids [16-19]. In this method, the global grid is maintained and new grid points are added at the features, yielding locally embedded patches. These patches may be aligned with the global grid [20], be rotated [16], or may be topologically dissimilar. This procedure maintains a basic accuracy on the global mesh and at the same time increases the otherwise reduced accuracy at features. Even though the scheme involves adding grid points (thereby increasing the total computational effort and required storage), they are only added where necessary, resulting in more efficiency than if comparable resolution were obtained by adding nodes globally.

Since the fine regions are embedded only locally, artificial internal boundaries with an abrupt grid spacing change result; special care must be taken there to account for this abrupt change and to assure both stability and conservation [17,21]. Coupling the global and embedded grids does not present a problem for aligned patches, since nodes are shared by the two grids. On the other hand, rotated or topologically dissimilar embedded regions require iteration between solutions on the two grids, with Dirichlet boundary conditions on each derived from the solution on the other. The chief disadvantage of grid adaptation by embedding is that the logic needed to implement this method is considerably more complicated than that for grid point redistribution. Processes are "turned on" or "turned off", depending on the type and location of various features. In addition to the detection process required of all adaptation schemes, some require that detected nodes be clustered before an embedded patch can be created [16].

The application of grid adaptation to some features points out a major deficiency of all the above schemes. To understand this, it is helpful to examine the concept of "features" more closely. Each feature can be characterized by its type, location, strength, orientation, and scale. The scale of a feature has special importance in classifying the feature as either collapsible or non-collapsible. The collapsible character of a feature implies that the scale is sufficiently small relative to the global scale such that the entire feature may be collapsed to a point, line, or surface with the implied associated physics taken into account. Examples of collapsible features are shocks and thin boundary layers.

In order to make the grid resolution fine enough to be able to capture the physics of a collapsible feature, grid point redistribution requires that a very large fraction of the available nodes be used at the feature, leaving other parts of the flow inadequately defined. In an embedded mesh scheme, many points would have to be added at the feature, resulting in a very inefficient calculation. It is interesting to note that as more points become available at a shock, the shock indeed becomes thinner (closer to the di-continuous physical solution). The gradient becomes larger (first difference remains the same), and even more grid points are required on the next pass. Thus, it is clear that grid adaptation alone is insufficient for resolving collapsible features.

Alternatively, if a feature is collapsible, adaptive equation techniques are very useful. In adaptive equation schemes, a special subset of the governing equations is used to adequately model the physics within the feature; shock fitting and recent strong inviscid/viscous interaction algorithms are examples of this approach. The chief advantage of these methods is that accurate solutions can be computed more efficiently than can a finite difference or finite volume method; this follows since the adapted equations are precisely the model for the local dominant physics.

For example, the "discontinuity" associated with a shock causes Gibbs' phenomenon in shock capturing methods, which assume the solution is smooth (representable by a Taylor series expansion) across the shock. Significant artificial viscosity levels are typically used to control the pre- and post-discontinuity oscillations. On the other hand, in shock fitting techniques [22-26], the discontinuity is modelled as a jump, with the solution being smooth on each side of, but not across the jump; this eliminates the oscillations since the solution is smooth where it is assumed smooth. The major drawback of shock fitting is that it typically requires that the shock lie along a grid line. As the shock moves, so does the grid. In addition, neither the number nor the existence of shocks is necessarily known beforehand, and this may lead to problems.

Another adaptive equation application is the popular inviscid/viscous interaction technique [27]. Here, it is known beforehand that the viscous effects are confined to thin layers near the boundary (or along the wake). An inviscid subset of the governing (Navier-Stokes) equations are solved external to the viscous region and the boundary/shear layer equations are solved where appropriate. The solutions are matched by using modified boundary conditions for the inviscid solver. These modified boundary conditions could be viewed as a "discontinuity" between the actual no-slip condition for the Navier-Stokes equations and the equivalent inviscid boundary conditions. Note that this technique requires that the viscous region be thin (so that the boundary layer assumption is not violated). If the assumption becomes invalid during the course of the solution development, equation adaptation can no longer be used, and the grid must be adapted to resolve the boundary layer.

Since the location of features are not always known a priori, the solution initially must be computed either without any adaptation or with adaptation at assumed feature locations. At a suitable time, features can be detected and adaptation begun. Recall that for a feature to be classified as collapsible and thus be a candidate for equation adaptation, the thickness (scale) of the feature must be less than one global cell width (a measure of the global scale). Generally this will not be the case when features are initially detected because the feature will be smeared over a number of global cells. The subsequent grid adaptation will eventually result in a feature which remains smeared over a few cells, but those adapted cells will now be small enough such that the overall scale of the feature will be less than one of the original global cells; thus equation adaptation can commence. It can be seen that although grid adaptation in general is inadequate for resolving the physics associated with collapsible features, it is a necessary first step in the classification of features as collapsible.

A new computer program, MITOSIS (MIT Optimizing Scheme for Integrating Systems), has been written to take advantage of the efficiencies in both adaptive equation and adaptive grid algorithms. An automatic algorithm is included which locates multiple features, determines their types, and subsequently which kinds of adaptation are most appropriate. Cell oriented schemes applied on a fixed global grid were chosen for convenient combination of the adaptation stategies with the unadapted base solver. Ni's finite-volume, Lax-Wendroff scheme with its multiple-grid accelerator [28] is used here as the basic solver. The embedded grid technique proposed by Usab and Murman [20] and a newly developed, floating shock fitting scheme are employed. The resulting scheme offers a high degree of computational efficiency with appreciable generality.

This paper is divided into two major sections. In the first section, the overall algorithm is briefly described and definitions used throughout the remainder of the paper are given. Following that, each major component of the scheme is described more fully, and specific options for each are discussed. The first section concludes with a discussion of the data structure which has been developed for easy implementation of both adaptation schemes. In the second section, two major examples using MITOSIS are discussed. The first example is for the one-dimensional Euler equations with both adapted grids and adapted equations. Computed results are shown for subsonic and transonic cases. The second example is for the two-dimensional Burgers equation with grid adaptation only. Computed results include a diffusion dominated case as well as a case with a curved discontinuity.

ADAPTIVE SOLUTION ALGORITHM

General Approach

Consider the integration of a system of hyperbolic partial differential equations in vector form:

$$U_t + F_x + G_y = H_x + I_y \qquad (1)$$

where the subscripts t, x and y denote differentiation with respect to time and the two space dimensions respectively. The state vector is given by U; the F and G are convective terms expressed in strong conservation law form whereas H and I are diffusive (non-conservative) terms. Both the Euler and Navier-Stokes equations can be written in this form. Interest is restricted to the resulting steady state behavior.

The approach taken here is to integrate the hyperbolic system to steady state by making available three different

integrators simultaneously. On a fixed global grid, the type of each cell (non-adapted, grid-adapted, or equation-adapted) is determined and the appropriate integrator is applied. Thus, all three cell integration schemes can be viewed as building blocks which can be assembled in any combination to yield the optimal adaptation for a given problem. Figure 1 depicts a typical flow field of interest, containing a shock, boundary layers, and a wake. A fixed global grid is superimposed. Cells which are unshaded represent those in which the basic (non-adapted) integrator is used, shaded cells are those in which grid adaptation is used, whereas cross-hatched cells contain collapsible features which are treated with equation adaptation.

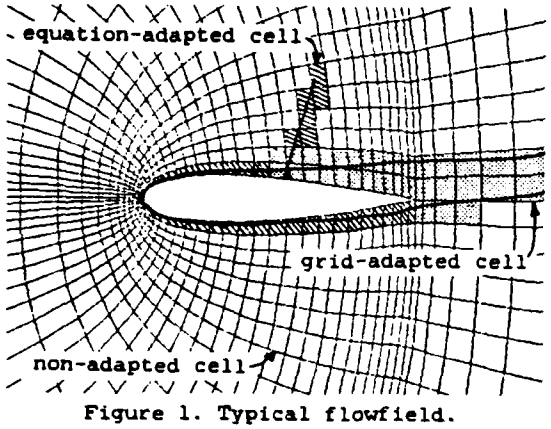


Figure 1. Typical flowfield.

A conceptual flowchart of such an integrating procedure (the MITOSIS program) is shown in figure 2. Each of the blocks in the flowchart will be briefly described in order to provide an overall view of the scheme; details follow in subsequent sections.

The process begins by initially assuming that there are no adapted regions present. A global grid is generated and the equations are integrated using the basic (non-adapted) integrator for all cells. Integration continues until features begin to form, as measured by a given level of convergence. At this point, a detection algorithm searches for the nodes whose variation (for example, gradient or truncation error) of a key variable (for example, density or entropy) are well above the average over the whole domain. All cells which are adjacent to those detected nodes (if any) are then divided, resulting in embedded grid cells.

Control then returns to the integrator, which now employs the appropriate combination of grid- and non-adapted algorithms. The detection algorithm is again applied after the features have reformed. On this and subsequent cycles, detected nodes are clustered so that those associated with a specific physical feature are treated as a group. The attributes (scale, location, orientation, etc.) of each cluster are determined; features (clusters) whose width scale is less than the global grid scale are further classified as collapsible. The clusters are individually compared with a library of standard patterns to determine if an alternate description for the local physics is available. Recognized features will subsequently be integrated with equation adaptation; the additional information required for equation adaptation is computed and any cells associated with the recognized feature which were divided during previous cycles are contracted (embedded cells removed). For unrecognized clusters, grid adaptation is again utilized.

Again, control returns to the integrator, which now employs any combination of the three integration schemes. Detection, clustering, attribution, and recognition are then repeated. Such cycling continues until no change in the adaptation strategies result from the above sub-processes.

Lastly, the equations are integrated until a final desired convergence level is reached.

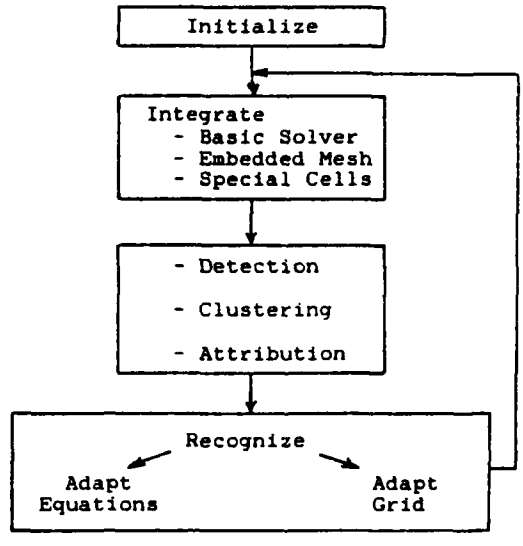


Figure 2. Conceptual flowchart.

Initialization

A body-fitted global grid (called the level 0 grid) which remains fixed throughout the solution procedure is generated by any convenient procedure (algebraic, PDE, etc.). Since the integration will involve a finite-volume formulation, there is no requirement that grid metrics be computed. The data structure, which consists of a cell oriented pointer system with solution variables

stored at the cell corners [20], is initialized. The data structure is described in detail below.

The basic integrator employs a multiple-grid accelerator for additional efficiency. This requires the generation of level -1 grids which are twice as coarse as the global (level 0) grid. Figure 3 illustrates the relationship of such cells. The level -1 grid was generated by eliminating every other line of the level 0 grid. Since cells on level 0 and -1 share nodes, the data structure (which exploits this fact) allows for an easy coupling of information on various multiple grid levels. Still coarser levels (-2, -3, etc,) are set up in a similar manner. Initially, all level 0 cells are considered basic (non-adapted) cells.

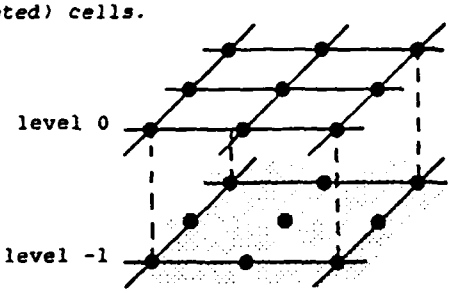


Figure 3. Cells on two multiple-grid levels.

Integration - Basic Solver

For non-adapted cells, integration of the governing equation is performed using Ni's multiple-grid algorithm [28]. This scheme is composed of two parts -- a finite-volume form of a standard single-step, Lax-Wendroff integration applied on a fine mesh, and a coarse grid accelerator which operates on residuals transported from the fine mesh solver. In both parts a "change" is computed in the center of each cell and then transferred to the adjacent nodes by means of "distribution formulae".

An essential point for present purposes is that Ni's scheme is cell-based; i.e., each cell can be integrated independently. The governing equations are approximated on the cell and the appropriate changes at each of the cell's nodes are computed. In this way, cells communicate with each other only through the dependent variable quantities at the shared nodes from the previous explicit pseudo-time step, or through changes at the nodes computed at the current pseudo-time level. This property is the basis of the data structure form.

Consider the fine mesh cell shown in figure 4. To calculate the "change" in the dependent variables at the center of this cell, the divergence theorem is applied to the convection terms of the governing equations, giving

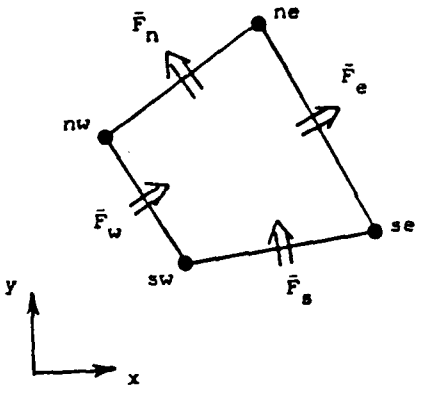


Figure 4. Basic cell with fluxes.

$$\Delta U = \frac{\Delta t}{\Delta V} \left[\bar{F}_w - \bar{F}_e + \bar{F}_s - \bar{F}_n \right] \qquad (2)$$

where $\bar{F}$ denotes the contravariant flux through each given face, Δt denotes the pseudo-time step, and ΔV the cell volume. The contravariant fluxes are computed by trapezoidal integration along each cell face; for example, the contravariant flux through the eastern face is given by

$$\bar{F}_e = \frac{F_{se}+F_{ne}}{2}(y_{ne}-y_{se}) - \frac{G_{se}+G_{ne}}{2}(x_{ne}-x_{se}) \qquad (3)$$

Equations (2) and (3) together are called the cell flux balance.

The distribution formulae serve to transfer this "change" from the center of the cell to the four corner nodes. The formulae are derived from the first two terms in a Taylor series expansion of U (with respect to time), and are given by

$$\delta U_{\substack{sw \\ se \\ ne \\ nw}} = \frac{1}{4} \left[\Delta U \mp \frac{\Delta t}{\Delta x} \Delta F \mp \frac{\Delta t}{\Delta y} \Delta G \right] \qquad (4)$$

where

$$\Delta F = \frac{\partial F}{\partial U} \Delta U \quad \text{and} \quad \Delta G = \frac{\partial G}{\partial U} \Delta U \qquad (5)$$

are the unsteady fluxes based upon the Jacobians of F and G evaluated at the center of the cell. The first term in equation (4) (ΔU) is the first-order-change-in-time for the Taylor series expansion while the last two terms represent a second-order-change-in-time which is necessary for stability. These terms bias the distribution of the "change" in the windward direction, which is somewhat similar to the stabilizing effects of upwind differencing [29].

Integration of the diffusive terms is performed by using a forward-time, centered-spaced scheme applied on staggered cells centered around nodes. One such staggered cell is shown by dashed lines in figure 5.

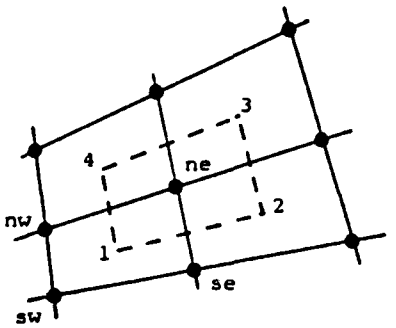


Figure 5.
Staggered cell for diffusive terms.

Each base cell (for example, cell SW-SE-NE-NW) contributes a first difference at its own cell center (for example, at node 1) toward the second derivative at the base node (for example, node NE). These contributions are calculated by centered, first differences in each base cell and are transferred to the base node by means of the diffusive distribution formulae

$$\delta U_{\substack{sw \\ se \\ ne \\ nw}} = \frac{1}{4} \left[\begin{matrix} - \\ + \end{matrix} \Delta H \begin{matrix} + \\ - \end{matrix} \Delta I \right] \qquad (6)$$

In practice, equations (4) and (6) are combined and evaluated simultaneously. Since only the steady state solution is of interest, the first order time accuracy of the forward-time, centered-space scheme is of no consequence.

In cases for which additional artificial viscosity is required, a spatially first order accurate smoothing term

$$\sigma \Delta x \, (U_{xx}+U_{yy})$$

is added to the right hand side of the governing equations. In the current scheme, the above terms are combined with the diffusive terms in equation (1), resulting in an effective viscosity coefficient with two contributions -- the true viscosity and the (varying) smoothing viscosity. Though implemented differently than the smoothing suggested by Ni, the effect of the two formulations is identical. As before, the effective diffusion is evaluated simultaneously with the convective distribution.

Consider now the coarse grid cell depicted in figure 6. The "change" at the center of this cell could again be performed by a flux balance, but with less accuracy than for the fine mesh flux balance due to the larger spatial discretization. To circumvent this problem, Ni uses the multigrid concept of "transporting" the changes previously calculated from the finer meshes [30]. The residual transfer then results in fine mesh accuracy. The transportation operation is simple to implement, since in the previous sweep on the fine mesh (shown by dashed lines in figure 6), the "changes" at node C were computed and stored; thus only access to this node is required. Note that the coarse grid scheme has no effect on the final, converged results since it operates on fine mesh residuals which vanish at convergence.

The distribution of the convective change on the coarse grid is again accomplished by using the distribution formulae (equations (4) and (5)). Since convective terms dominate over diffusive ones in problems of interest, the latter are neglected on the coarse mesh, as suggested by Johnson [31]. No smoothing is applied in the coarse mesh accelerator.

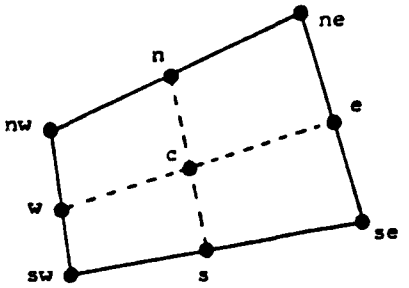


Figure 6. Coarse grid cell.

One cycle of the Ni multiple-grid integration scheme is shown in figure 7. It consists of the following operations:

1. Initialize: Changes in the dependent variables stored at each node are set to zero at the beginning of each multiple-grid cycle.

2. Flux balance and distribute on level 0: Cell by cell operations are performed as described above.

3. Apply boundary conditions on level 0: Characteristic boundary conditions are applied at each boundary node. The boundary condition formulation treats characteristic waves which enter and exit the computational domain differently, similar to Chakravarthy's development [32]. For those characteristics which exit the domain, it is assumed that the change of the characteristic variable is properly predicted by the distributuion formulae. The characteristic variables corresponding to waves which enter the domain are kept unchanged from their previous values. Finally, the changes in the characteristic variables from above are recombined to give the conservation variable changes.

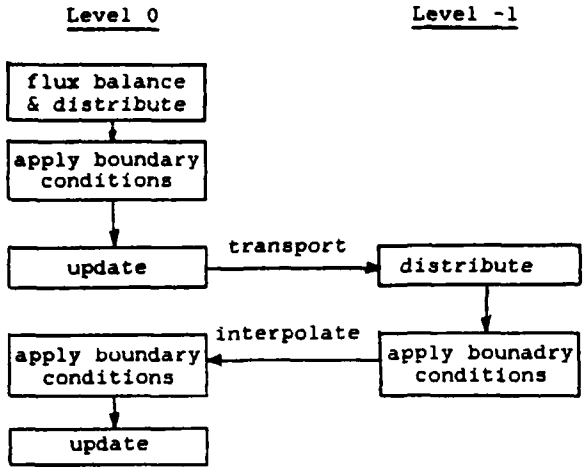


Figure 7. Ni multiple-grid cycle.

4. Update on level 0: The change at each node is added to the dependent variable at the node. Since time accuracy is not of interest, the time step for this update is computed locally based upon a global CFL number and the local solution behavior. The convergence to steady state is measured by the largest change in δU at any node on level 0.

5. Transport from level 0 to level -1 and distribute: Cell by cell calculations are performed as described above.

6. Apply boundary conditions on level -1: Same as in step 3.

7. Interpolate from level -1 to level 0: Changes at the side and center nodes must be interpolated based upon the recently computed corner changes. This is accomplished by bilinear interpolation. A shifted "distribution formula" which does not generate the smoothing errors associated with simple interpolation is another possibility; this however requires significantly more operations than the bilinear scheme and has not yet proven to yield superior convergence rates for all cases.

8. Apply boundary conditions on level 0: Again the procedure in step 3 is applied, this time to ensure that values interpolated onto boundaries satisfy the boundary conditions.

9. Update on level 0: The level -1 changes determined either from the distribution in step 5 or the interpolation in step 7 are added to the solution dependent variables.

The use of Ni's integration scheme as the non-adapted cell integrator is not imperative; the only requirements are that the scheme be explicit and expressible on a cell-by-cell basis. As will be seen however, the similarity of the fine and coarse solvers in Ni's scheme greatly simplifies the formulation of the adaptive grid integration.

Integration - Embedded Grid Procedure

For grid-adapted cells, the embedded mesh procedure of Usab and Murman [20] is used. This is a straightforward combination of the fine and coarse techniques used for non-adapted cells. Consider the geometry shown in figure 8. The unshaded cells are fine whereas those shaded are coarse (since they coexist with finer cells). The appropriate fine or coarse operation outlined above is applied to each cell. Thus, two consecutive cells on the same level may be integrated by different procedures. Away from the fine/coarse interface, the integration proceeds as usual, with one more multiple-grid level in the embedded region.

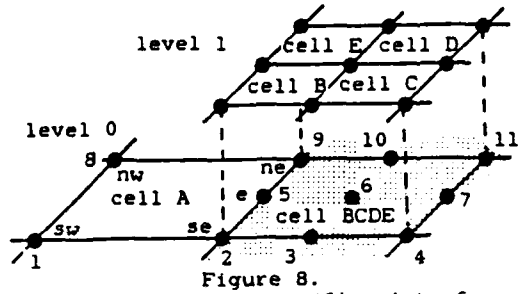


Figure 8.
Embedded grid -- coarse/fine interface.

Using the multiple-grid accelerator to couple the global and embedded meshes was first suggested by Brandt [33] and has been implemented by Brown for the full potential equation [34]. With this technique, waves can propagate through the embedded regions at coarse grid speeds. Usab has shown that this coupling results in convergence rates which are as fast as coarse-grid-alone solutions [20]. This is significant when one considers the consequences of simply coupling global and embedded regions at the interface [21]. In the latter technique, wave propagation is restricted to the embedded (fine) grid speed, resulting in slower convergence rates.

Points at the edge of the embedded domain must be carefully treated in order to maintain global conservation and computational stability. In the present scheme, nodes 2, 5, and 9 are considered part of the fine domain. Thus the flux balance and distribution formulae can be applied as usual to the level 1 cells (B, C, D, and E). Although the changes in the dependent variables are computed at nodes 2, 5, and 9 due to cells B and D, the changes must not be applied when operating on level 1. Instead, they must be stored and applied only after the (explicit) flux balance has been performed on cell A.

Since cell A is a fine cell on level 0, the appropriate integration consists of a flux balance and distribution. Each of these steps must be modified due to the presence of node 5. For this cell, the contravariant fluxes, $\bar{F}_s$, $\bar{F}_n$, and $\bar{F}_w$ are given in equation (3), but the flux through the eastern edge is given by

$$\bar{F}_e = \frac{F_{se}+F_e}{2}(y_e-y_{se}) - \frac{G_{se}+G_e}{2}(x_e-x_{se}) + \frac{F_{ne}+F_e}{2}(y_{ne}-y_e) - \frac{G_{ne}+G_e}{2}(x_{ne}-x_e) \quad (7)$$

If one assumes that node 5 is at the midpoint of the eastern edge (as it is by construction), then equation (7) takes the simpler form

$$\bar{F}_e = \frac{F_{se}+2F_e+F_{ne}}{4}(y_{ne}-y_{se}) - \frac{G_{se}+2G_e+G_{ne}}{4}(x_{ne}-x_{se}) \quad (8)$$

The distribution formulae given by equation (4) still are valid for cell A. However, the change at the center of the cell must also be distributed to node 5. Presently, this is accomplished by averaging the distribution to nodes 2 and 9, or in general

$$\delta U_{\substack{e\\ \diamond}} = \frac{1}{4}\left[\Delta U \pm \frac{\Delta t}{\Delta x}\Delta F\right]$$

and (9)

$$\delta U_{\substack{s\\ \diamond}} = \frac{1}{4}\left[\Delta U \mp \frac{\Delta t}{\Delta y}\Delta G\right]$$

The scheme maintains global conservation since the flux balances exactly cancel at the fine/coarse interface, and no additional mass is created as a result of equation (9).

Cell BCDE is treated the same as any other coarse cell interior to the embedded region. This yields an apparent inconsistency at nodes 2, 5, and 9 due to the absence of a coarse cell underlying cell A. At convergence however, the residuals transferred to BCDE do vanish as do the inconsistencies.

Integration - Special Cells

Recall that equation adaptation is used in cases where the local dominant physics can be applied to a collapsed region to yield a discontinuous solution. For such cells, the integration scheme must integrate the basic governing equations on each side of the discontinuity (which is free to move to any location in the cell) and also be capable of computing the discontinuity location and strength consistent with the collapsed physical model. The present scheme accomplishes this in two steps. First, the equations are integrated over the cell, with an assumed discontinuity location and strength. This includes a flux balance and application of the distribution formulae. The second part consists of updating the discontinuity location and strength. This two step procedure, which was first suggested by de Neef [22], is consistent with the explicit nature of the overall scheme.

For illustrative purposes, consider the case of a shock in a quasi-one-dimensional cell, as shown in figure 9. For an assumed shock location and strength, the flux balance for this cell may be written as

$$\Delta U = \frac{\Delta t}{\Delta x}\left[(F_1-F_2) + (F_2-F_3) + (F_3-F_4)\right] \quad (10)$$

where the first and last terms in parentheses represent the flux balance to the left and right of the shock, respectively. The flux balance (F_2-F_3) represents the flux balance across the shock, which is zero for equations written in conservation form. Equation (10) may be rewritten in the simpler form

$$\Delta U = \frac{\Delta t}{\Delta x}\left[(F_1-F_4) + (F_3-F_2)\right] \quad (11)$$

where the first parentheses represents the flux balance for a non-adapted cell and the latter may be interpreted as a correction due to the shock jump. The one dimensional form of the distribution formulae (4) are then applied to the change given by equation (11). The Jacobian required in the distribution is computed based on the average value of U over the cell.

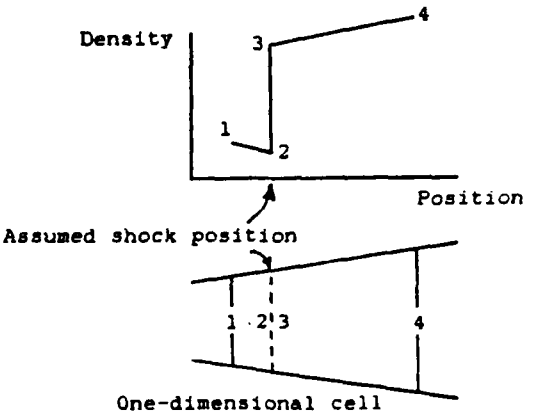


Figure 9. Shock fitting nomenclature.

To compute the proper shock location and strength, the characteristics (Riemann variables) which coalesce to form the shock are integrated to the new time and the appropriate shock speed and strength determined. Figure 10 shows time versus position away from the shock. The horizontal axis represents the time level from the previous multiple-grid cycle. The characteristics a, b, and c are downstream running from the supersonic flow, while d is

the upwind running characteristic from the subsonic flow. After point e is located (based upon the maximum Δt allowed by the CFL condition and the cell size), the intersections of the four characteristics emanating from e and the horizontal axis are determined. The characteristic variables at those points are determined (by interpolation of the conservation variables), and integrated forward to point e. The a, b, and c characteristic variables uniquely determine the conditions just upstream of the shock, and hence its strength. With this strength and the known characteristic variable (from d) just after the shock, the shock speed can be determined iteratively. The speed is then integrated to give a new shock location. For the converged solution, the shock speed vanishes, and thus the shock remains stationary.

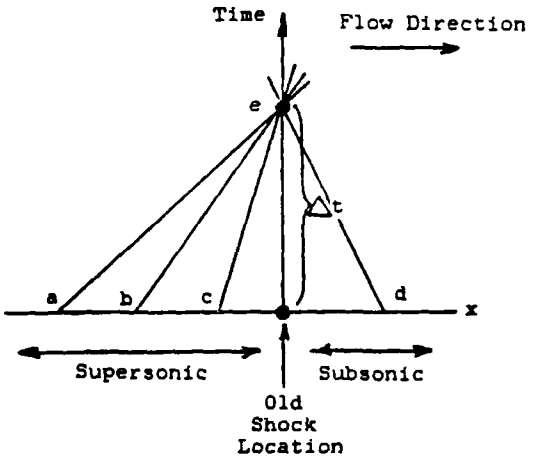


Figure 10. Characteristic integration.

The shock speed and strength are only computed based upon the fine grid and are thus frozen throughout the multiple-grid acceleration cycle. By this means, not only is the fine grid accuracy maintained, but also the multiple-grid cycle converges as rapidly as if the shock was absent. This result is very similar to the results stated by Boerstoel and Kassies [35].

Care must be taken when a solution discontinuity passes from one cell to another. The dependent variable at the node over which the discontinuity passes must then be adjusted and a new cell designated as an equation-adapted cell. The change in dependent variables which results from a discontinuity passing a node must not be included however in the change transported to coarser multiple-grid levels, since a very large change (i.e., the discontinuity jump) is inconsistent with changes resulting from the flux balance and distribution.

Feature Detection

Detected nodes are points in the computational domain at which the "variation" of some "key variable" is large, indicating that adaptation is necessary. Determining which nodes are detected is a two step process. First, the variation must be computed at each node; then, a threshold must be set to separate the signal (detected nodes) from the background noise.

The "variation" can be computed in a variety of ways. Popular choices include gradient [3], Laplacian (second derivative) [7], and truncation error [18]. To be useful, the gradient calculation should be performed in the computational domain (a first difference in the physical domain) to account for the effect of the smaller grid spacing associated with adaptation. Gradient and Laplacian are relatively simple to compute and are therefore the most popular. Berger [16] has developed a method which directly measures the truncation error through the use of a Richardson extrapolation. This method is more complicated to compute than the others but is claimed to be problem and integrator independent.

The choice of the "key variable" is apparent for scalar equations. However, in problems involving a set of governing equations, the choice of the "key variable" is not obvious. For example, choices for the Euler equations include density, momentum, velocity, vorticity, and entropy. It is possible that there may be more than one "key variable" for certain problems. The important consideration in choosing the key variable(s) is that it(they) vary in all the expected feature types; the most appropriate "key variable" for the Euler equations has not yet been identified.

The second step of detection is the determination of an appropriate threshold, above which points are detected. The thresholding algorithm should be general enough to work under a variety of circumstances. If the threshold is too low, too many points are detected, resulting in wasted adaptation. On the other hand, if the threshold is set too high, important features may be missed.

Figure 11 shows the effect of selecting various thresholds; the fraction of points with "variation" above the threshold is plotted versus threshold level. As can be seen, over a large range of threshold, the fraction of thresholded points is constant. However, if the threshold is reduced below that value, nodes in the "noise" are accepted. Thus the threshold is set to find the "knee" in such a plot.

There are cases which contain no features and thus no adaptation is desired. In these, the threshold plot will not have a "knee" and additional constraints are necessary to properly set the threshold. Adequate constraints are easily formulated by setting a lower bound on the threshold (typically 20 percent above the average

"variation") as well as a limit on the number of nodes detected on any pass (typically 30 percent). In all computed cases shown in the results section, these values were used and found to be sufficient.

In an evolving field computation, features may move or be narrowed by previous adaptations. Thus, a "contraction threshold" is set so that unnecessary adaptation may be eliminated. This value is typically 25 percent of the average "variation".

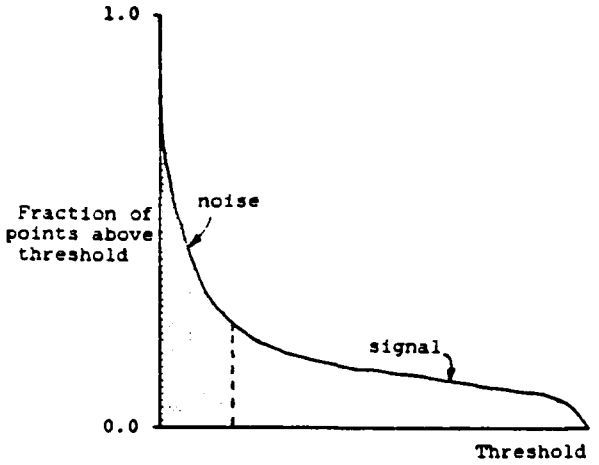


Figure 11. Signal/noise discriminator.

Clustering

Detected nodes which are contiguous are assumed to be part of a single physical phenomenon, and thus should be clustered together. Standard hierarchical and non-hierarchical clustering techniques [36] are generally iterative in nature, requiring many passes through the detected nodes. This is primarily due to the fact that these techniques have no a priori basis for measuring contiguousness.

Since the grid can be used to measure contiguousness, a new clustering technique has been developed which uses grid structure information. In the new technique, which can cluster the detected nodes in a single pass, the cells are scanned and the following operations are performed:

- If no nodes have been previously assigned to any cluster, then--

  If one or more of the nodes are detected, associate this cell with a new cluster, mark all the nodes as members of the cluster, and continue on to the next cell.

  Otherwise, continue on to the next cell.

- If all assigned nodes belong to the same cluster, associate the remaining nodes with the cluster, flag the cell as member of the cluster, and continue on to the next cell.

- In the event that more than one cluster is associated with this cell, then merge all of the pertinent clusters into one, and continue as if all the nodes belonged to the same cluster.

From the above, it can be seen that the clustering is accomplished in a single pass, with the exception of the merging operation which simply requires reassigning cluster numbers. The resulting clustering algorithm has been found to execute very rapidly, resulting in insignificant computer times when compared with the integration time.

Attribution

Once clusters have been formed, the attributes of the cluster must be determined as a first step in the recognition process. Initially, the orientation of each cluster must be determined. By comparing the orientation and location of each cluster, it is possible to determine if more than one contiguous physical feature is likely to be present. Since, in general, features will not abruptly change orientation, any significant orientation change is used to break a cluster into sub-clusters. For example, this algorithm would break up the intersection of a shock and a boundary layer into four features (figure 12).

Once features have been properly subdivided, the "width" of the feature normal to the feature orientation can be measured at various locations. If these widths are smaller than the local global grid spacing, the feature is designated as "collapsible". Note that the width is measured only with respect to detected nodes, and thus for a discontinuity such as a shock, it continually becomes thinner with repeated adaptations.

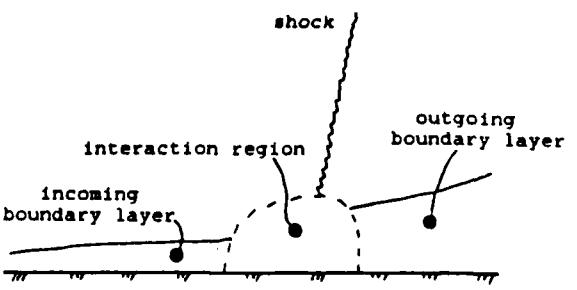


Figure 12. Four features from one cluster.

Recognition

Collapsible features are now compared with standard patterns to determine if an associated set of local physics is consistent with the feature. Such a pattern for a shock may assume the form: incoming, supersonic normal Mach number; outgoing, subsonic normal Mach number; entropy increase; etc. Recognized features then are tagged as to type.

Grid Division and Contraction

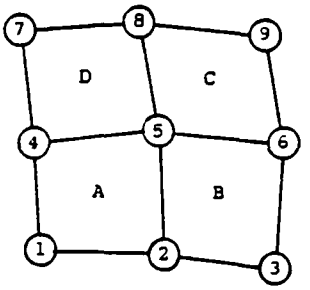
Processes are needed to accomplish grid division and contraction for both grid and equation adaptation. In grid adaptation, the cells surrounding a detected node are subdivided, as long as the cell to be divided is not already at the edge of an embedded region. Such a restriction is necessary since the cell just outside the embedded region would otherwise be described by four or more nodes along a single edge; this eventuality could cause considerable difficulties in the basic and embedded grid solvers.

Data Structure

In each algorithm described above, the computations are performed cell by cell. There is no requirement that the nodes be stored in any specific order as long as those associated with each cell are known. This, coupled with the complex arrangement of nodes which may result from successive grid adaptations, leads to the requirement for a flexible data structure. Usab has developed a pointer system [37] which offers such flexibility.

The data structure is shown in figure 13. Node information is stored in a single array with the arrangement of node numbers arbitrary. For each node, its location, the dependent variables, and the changes in those dependent variables are stored.

Connectivity or cell arrays contain the numbers of the nine nodes associated with each cell (0 if a node does not exist). Thus, all interior nodes are accessed by four cells on level 0 (the global level). Because of the multiple-grid structure, each node is actually accessed on each level of which it is a member. Also included for each cell is a special word which contains information about the cell's location relative to the domain boundary and edge of an embedded region; this word also includes a tag indicating the type of integrator (non-adapted, grid-adapted, or equation-adapted) used for the cell. This cell information is stored in one long array, organized so that all cells of the same level are stored contiguously. A set of level pointers are used to indicate the first and last cell on each level. This type of organization makes even complicated global topologies as shown in figure 14 very easy to implement.



Nodes:

1	2	3	4	5	6	...

Cells:

	A	B	C	D	...
sw	1	2	5	4	
se	2	3	6	5	
nw	5	6	9	8	
ne	4	5	8	7	

Figure 13. Data structure.

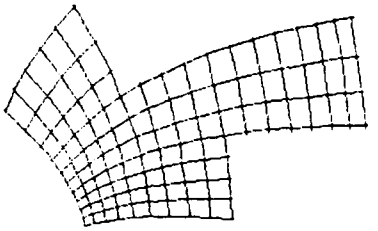


Figure 14.
Global grid with complex topology.

Two additional arrays are required to complete the data structure. The first is a boundary condition array, which contains nodes numbers for all boundary nodes, the adjoining cells used in boundary condition calculations, and the boundary condition type (solid wall, free stream, etc.). For special cells, the other array contains feature types and discontinuity values.

Though the data structure may seem complicated, utility routines have been written which make grid manipulation simple. For example, a grid division routine creates the required nodes and cells and readjusts all indicators which contain edge-of-embedded region information. Such routines are problem independent.

ONE-DIMENSIONAL EULER EQUATION

The first set of results illustrate both grid- and equation-adaptation applied to the quasi-one-dimensional Euler flow through a converging-diverging nozzle. Cases with and without shocks are considered.

Governing Equation and Boundary Conditions

The one-dimensional Euler equations in conservation form are given by

$$U_t = F_x + \frac{G}{A} A_x \qquad (12)$$

where

$$U = \begin{vmatrix} \rho \\ m \\ E \end{vmatrix} \qquad F = \begin{vmatrix} m \\ \frac{3-\gamma}{2}\frac{m^2}{\rho} + (\gamma-1)E \\ \frac{1-\gamma}{2}\frac{m^3}{\rho^2} + \gamma\frac{Em}{\rho} \end{vmatrix}$$

$$G = \begin{vmatrix} m \\ \frac{m^2}{\rho} \\ \frac{1-\gamma}{2}\frac{m^3}{\rho^2} + \gamma\frac{Em}{\rho} \end{vmatrix}$$

The three equations represent conservation of mass, momentum, and internal energy, respectively. The source terms are due to the area change along the duct. Characteristic boundary conditions applied at the inlet and exit of the duct assume those characteristics which exit the domain are correctly predicted by the distribution formulae, while for those entering, the characteristic variables remain unchanged. The cases shown all have a subsonic inlet at which two boundary conditions are prescribed and a subsonic exit for which only one is given.

Computed Results

The first case is a choked flow, with a shock in the expanding section. Figure 15a shows the duct geometry with the global grid superimposed. The global grid is equally spaced here, although this is not a requirement. The computed axial Mach number distribution is plotted in figure 16a. Note that the captured shock extends over approximately three cells. The entropy change distribution (figure 17a) shows that the added artificial viscosity that was necessary to control oscillations near the shock results in a significant entropy rise in adjacent regions and hence the incorrect placement of the shock. As a result of capturing the shock, the computed entropy rise across the shock is approximately 21 percent too small as compared with the analytical value associated with this incident Mach number. The mass flow rate distribution (figure 18a), indicates that the added artificial viscosity causes appreciable mass creation and destruction before and after the shock; the peaks do not cancel exactly in this case, resulting in a 0.6 percent residual mass flow rate error after the shock.

Since the added artificial viscosity is proportional to the local mesh spacing, it is expected that finer mesh resolution will result in both better shock resolution and placement. Figure 15b depicts the geometry after dividing each global cell into quarters. The resulting Mach number, entropy, and mass distributions are shown in figures 16b, 17b, and 18b, respectively. Note that the shock has moved downstream relative to the non-adapted solution as a

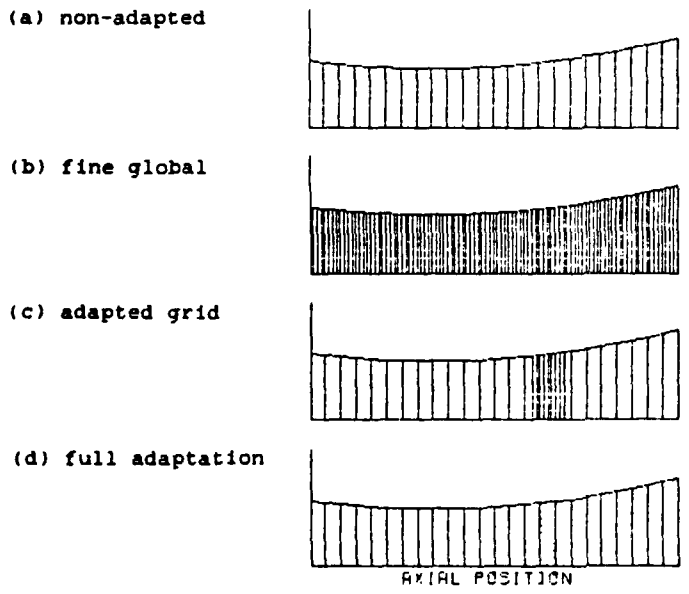


Figure 15. Grid distribution.

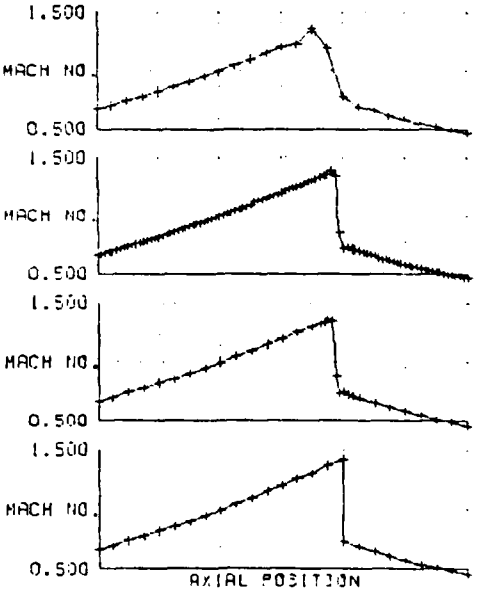


Figure 16. Mach number distribution.

result of the lesser entropy rise ahead of the shock which followed from the smaller added artificial viscosity. The computed entropy rise across the shock is now within 4 percent of its theoretical value while the mass flow rate error has been reduced to 0.4 percent.

Figures 15c, 16c, 17c, and 18c contain results for grid adaptation alone. Note that the shock width has been reduced, although it still extends over three cells. The artificial viscosity in the large (non-adapted) cells ahead of the embedded region again causes the shock to be incorrectly located even though the entropy rise across the captured shock is now only about 6 percent low. This error, which is approximately the same as the fine global grid case (b), is appreciably smaller than that for the non-adapted case (a).

Full (i.e., both grid and equation) adaptation results are shown in figures 15d, 16d, 17d, and 18d. Here, both the shock position and entropy rise are correctly predicted. Note also that since the shock is fit (not captured), virtually no mass flow error exists. The results were obtained from the following sequence of operations: first, a non-adapted solution was computed until a shock formed. Since the detected cluster was not collapsible (spread over 2 global cells) grid-adaptation was performed. The integration continued until the effects of adaptation had been established. The detected nodes then extended over two fine (adapted) cells with a total width less than one global cell. Equation adaptation was initiated and previously embedded cells were deleted. After advancing the solution and applying detection again, no further adaptation was required. The artificial viscosity parameter was reduced (since it is unnecessary to fit the shock) and the solution advanced to final convergence. The ability to decrease the artificial viscosity parameter is important in obtaining an overall accurate result.

Figure 19 shows convergence histories for each of the above four cases. The logarithm of the average change in momentum over the whole domain is shown as a function of the number of multiple-grid cycles. Figure 20 similarly shows the convergence histories versus work units, which are the accumulated CPU time on the system clock normalized by the CPU time required for the non-adapted case. This work measure includes any time used by the detection, adaptation, etc. algorithms. The dotted lines in the two figures show the convergence of the non-adapted calculation. Note that the fully embedded grid (dot/dashed line) requires more than 2.5 times as many cycles with an even more dramatic work difference due to the lengthier calculation required for the larger number of grid points.

The adapted grid results (dashed line) shows some interesting results. The convergence history is identical to the non-adapted case until the first grid adaptation, corresponding to the abrupt rise in the residual. In the work plot, the rise occurs over a relatively small region,

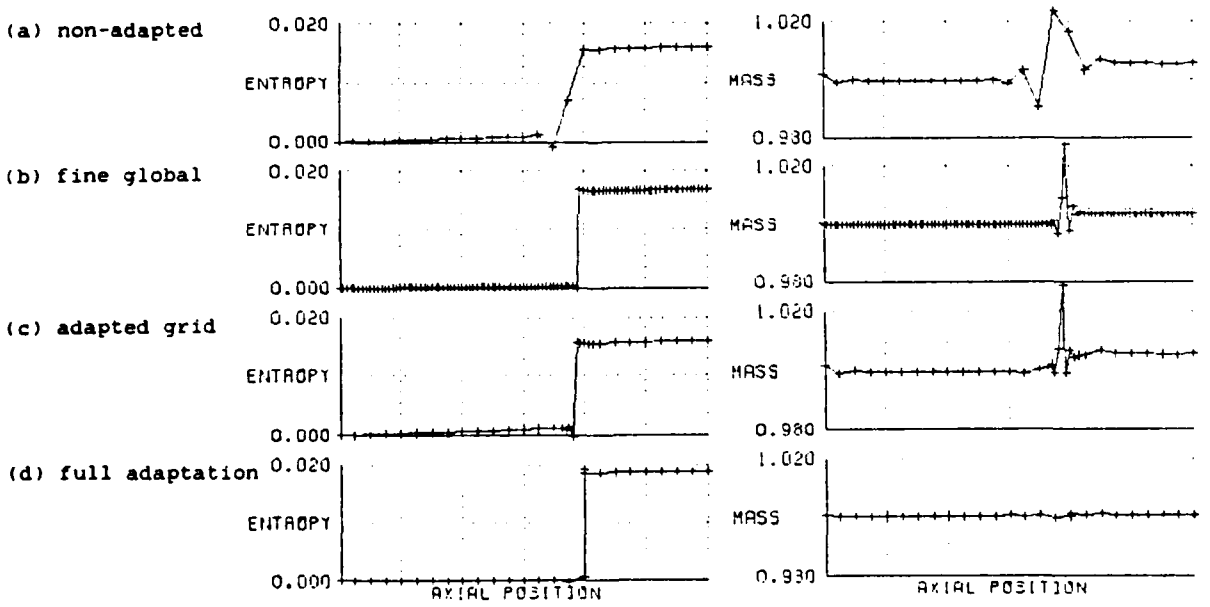


Figure 17. Entropy distribution. Figure 18. Mass flow rate distribution.

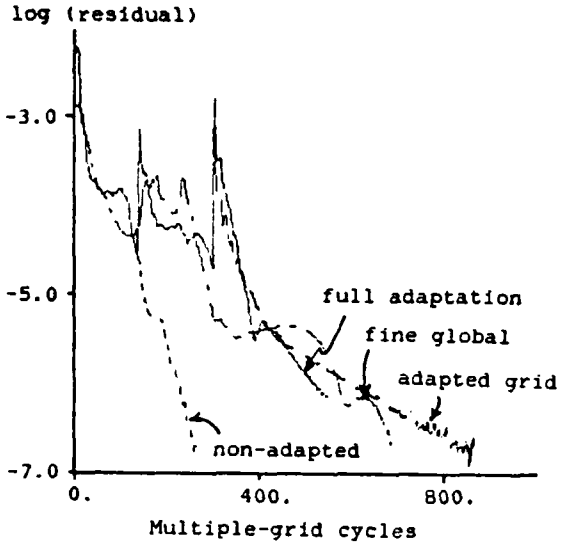


Figure 19.
Convergence histories - multiple-grid cycles.

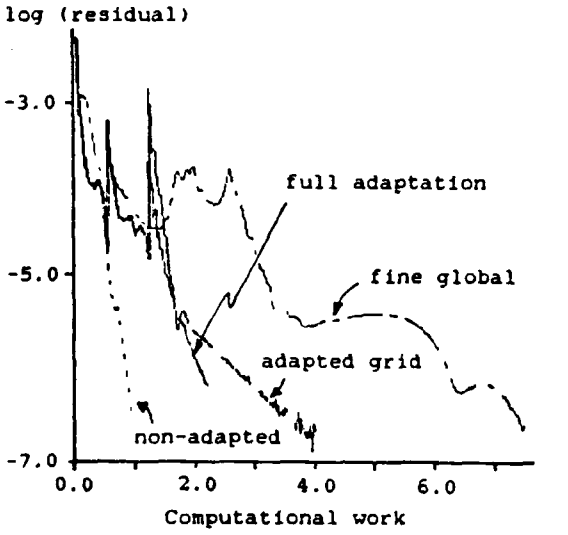


Figure 20.
Convergence histories - computational work.

indicating that the adaptation calculations are very inexpensive compared to the integration time. The second abrupt rise corresponds to the second grid adaptation. The differences in the slopes between the cycle plot (figure 19) and the work plot (figure 20) is due the presence of more nodes after adaptation. The fully-adapted solution is shown by the solid line and again shows two abrupt rises, the first due to the grid adaptation and the second due to the equation adaptation. Again the adaptation time is negligible compared with the integration time. The more negative slope in the work plot (as compared with grid adaptation alone) after equation adaptation is due to the reduced number of computational nodes.

A fifth case was computed with an a priori embedded grid identical to the final grid from the adapted grid results, yielding the same solution (as expected). The a priori embedding required 6 percent fewer multiple-grid cycles, since the residual rises associated with adaptation were not present. However, the total computational work required for the a priori embedding was 4 percent greater, owing primarily to the less expensive computation per cycle before adaptation; this is analogous to the efficiency gains resulting from coarse/fine sequencing.

To demonstrate that a specified level of residual is an appropriate measure of feature formation, figure 21 shows the development of the Mach number distribution versus residual. After the residual has decreased about two decades, the shock location and strength are fairly well established, with only small adjustments over the last two decades. Since adaptation changes the fine structure near the feature anyway, significant savings can be realized if the adaptation is performed after the residual drops only two decades.

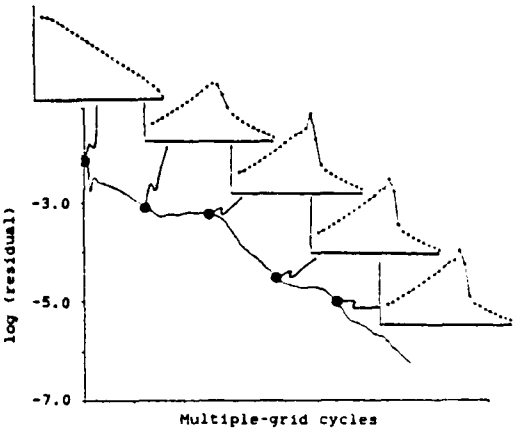


Figure 21. Mach number distribution for selected residual levels.

The second test case assumes the same geometry, but this time with insufficient back pressure to choke the flow. The results are shown in figure 22. The detection process correctly found no nodes and adaptation was not performed.

A final one-dimensional test case considered the nozzle with a wavy wall upstream of the throat. By this means, two distinct features were included (one at the shock and one due to the wavy geometry) which were found and treated independently. The shock was automatically recognized and the equations adapted for shock fitting; the geometrically generated feature was detected but not recognized, resulting only

in grid adaptation. These results, which are shown in figure 23, demonstrate the flexibility and generality of the present method.

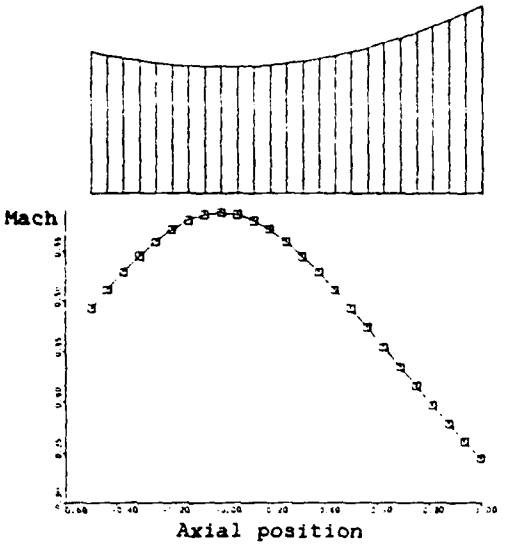


Figure 22. Unchoked nozzle.

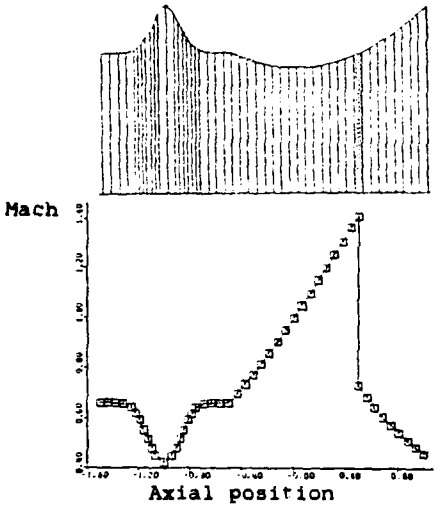


Figure 23. Multiple features.

TWO-DIMENSIONAL BURGERS EQUATION

Similar concepts have been used to extend the grid adaptation procedure to two-dimensional models based on the Burgers equation, offering two new complications. In contrast to the one-dimensional case for which clustering is almost trivial and the coarse/fine interface is relatively easy to implement, a second dimension introduces appreciable complexity. The second complication arises from the presence of diffusive terms. If modeled by a centered second difference split between two different scale cells, the coarse/fine interface condition must be carefully modified. The equation is a good model equation for the Navier Stokes equations excepting for its scalar nature. The latter is the major justification for using Burgers equation, namely rapid computation with the essential contents of a two-dimensional viscous description included.

Governing Equation and Boundary Conditions

Burgers equation in two dimensions and conservation form is given by

$$U_t + \frac{1}{2}(U^2)_x + \frac{1}{2}(U^2)_y = \mu[U_{xx}+U_{yy}] \quad (13)$$

Characteristic boundary conditions have been applied as derived from the convective terms. It is assumed that the convective terms dominate at entrance and exit regions. No added artificial viscosity is used in any of the computed two-dimensional results.

Computed Results

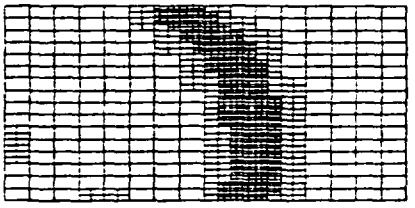
The non-linear convective terms in equation (13) allow for a discontinuity; a test case was computed with such a curved "shock". The final adapted grid and a contour plot of U are shown in figure 24. The adapted equation algorithm is not yet included in the two-dimensional code and therefore the discontinuity has been captured, but not fit. The irregular shape of the embedded region necessary for this feature is an essential capability of the algorithm for a still broader class of descriptions. The flexibility of the approach and the inherent data structure make the computation of this case no more complicated than any other.

As a demonstration of the effect of grid adaptation, a horizontal cut has been made across the domain, and both non-adapted and grid-adapted results are shown in figure 24c. It is clear that the grid adaptation yields much better resolution of the discontinuity.

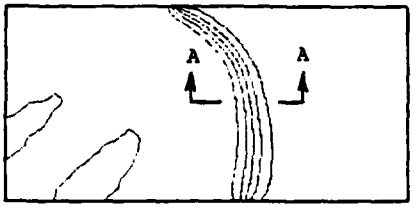
The second two dimensional case considers a situation in which diffusion is important. A perturbation at the lower left corner of a rectangular domain (figure 25) propagates up and to the right and at the same time diffuses. Again it is of some importance that the embedded region is irregularly shaped. Note that the solution smoothly traverses the edge of the embedded region. Even though the diffusion is important here (approximately equal to a Reynolds number of 10 based upon the domain length), the characteristic boundary conditions based only upon convective terms

seem to be effective and proved satisfactory since a singular perturbation (boundary layer) was not present.

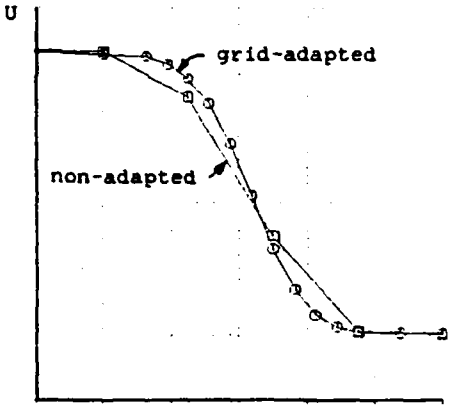
Currently, program MITOSIS is being extended to two-dimensional Navier-Stokes equations with both grid and equation adaptation. Although quite general and in principle extendable to three dimensions, such efforts will require some care.



(a) computational grid

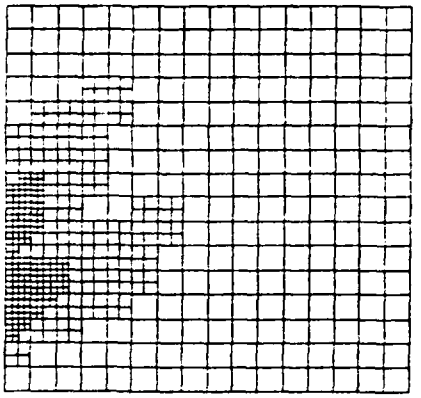


(b) contours of U

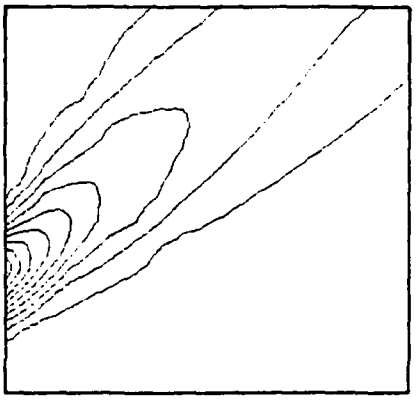


(c) Section A-A

Figure 24.
Burgers equation - curved "shock".



(a) computational grid



(b) contours of U

Figure 25.
Burgers equation - diffusive field.

CONCLUSIONS

- A single algorithm has been developed which combines adaptive grid and equation techniques, yielding both significant time savings and accuracy improvements.
- Shock fitting can be combined with a Lax-Wendroff scheme (including mutiple-grid acceleration) to obtain converged solutions with a significantly reduced requirement for artificial viscosity.
- Irregular embedded grid regions which track features are an essential component for multi-dimensional problems.
- The compu tional work (both direct and indirect) associated with adaptation is small compared with that saved by the increased efficiency of the adaptation.
- A new, single-pass clustering algorithm has been developed.
- A flexible data base system is essential for effective combination of adaptive grid and equation algorithms.

ACKNOWLEDGEMENTS

This work was supported by the Air Force Office of Scientific Research under grant AFOSR-82-0136, Dr. J Wilson technical monitor. The authors wish to thank Drs. E Murman and W Usab for their suggestions and enlightening conversations.

REFERENCES

[1] Anderson DA, "Adaptive Mesh Schemes Based on Grid Speeds", AIAA-83-1931-CP, July 1983.

[2] Gnoffo PA, "A Vectorized, Finite-Volume, Adaptive Grid Algorithm Applied to Planetary Entry Problems", AIAA-82-1018, June 1982.

[3] Greenberg JB, "A New Self-Adaptive Grid Method", AIAA-83-1934-CP, July 1983.

[4] Anderson DA and Rai MM, "The Use of Adaptive Grids in Solving Partial Differential Equations", Numerical Grid Generation, JF Thompson, Ed., Elsevier Science Publ., New York, 1982, pp 317-338.

[5] Babuska I and Rheinboldt WC, "Error Estimates for Adaptive Finite Element Computations", SIAM J. Numer. Anal., Vol 15, No 4, August 1978, pp 736-754.

[6] Davis SF and Flaherty JE, "An Adaptive Finite Element Method for Initial-Boundary Value Problems for Partial Differentail Equations", SIAM J. Sci. Stat. Comp., Vol 3, No 1, March 1982, pp 6-27.

[7] Dwyer HA, "A Discussion of Some Criteria for the Use of Adaptive Gridding", AIAA-83-1932-CP, July 1983.

[8] Eiseman PR, "Alternating Direction Adaptive Grid Generation", AIAA-83-1937-CP, July 1983.

[9] Harten A and Hyman JM, "Self-Adjusting Grid Methods for One-Dimensional Hyperbolic Conservation Laws", J. of Comp. Phy., Vol 50, 1983, pp 235-269.

[10] Hindman RG and Spencer J, "A New Approach to Truly Adaptive Grid Generation", AIAA-83-0450, January 1983.

[11] Mastin CW and Thompson JF, "Adaptive Grids Generated by Elliptic Systems", AIAA-83-0451, January 1983.

[12] Brackbill JU, "Coordinate System Control: Adaptive Meshes", Numerical Grid Generation, JF Thompson, Ed., Elsevier Science Publ., New York, 1982, pp 277-294.

[13] Dwyer HA, Smooke MD, and Kee RJ, "Adaptive Gridding for Finite Difference Solutions to Heat and Mass Transfer Problems", Numerical Grid Generation, JF Thompson, Ed., Elsevier Science Publ., New York, 1982, pp 339-356.

[14] Thompson JF, "The Construction and Use of Adaptive Grids", University of Tennessee at Knoxville Rept E01-1350-01-001-84, July 1983.

[15] Mastin CW, "Error Introduced by Coordinate Systems", Numerical Grid Generation, JF Thompson, Ed., Elsevier Science Publ., New York, 1982, pp 31-40.

[16] Berger MJ, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations", Rept. STAN-CS-82-924, Stanford University, August 1982.

[17] Berger MJ, "Stability of Interfaces with Mesh Refinement", ICASE Rept. 83-42, August 1983.

[18] Bolstad JH, "An Adaptive Finite-Difference Method for Hyperbolic Systems in One Space Dimension", Rept. STAN-CS-82-899, June 1982.

[19] Gropp WD, "A Test of Moving Mesh Refinement for 2-D Scalar Hyperbolic Problems", SIAM J. Sci. Stat. Comput., Vol 1, No 2, June 1980, pp 191-197.

[20] Usab WJ and Murman EM, "Embedded Mesh Solution of the Euler Equation Using a Multiple-Grid Method", AIAA-83-1946-CP, July 1983.

[21] Berger MJ and Jameson A, "Automatic Adaptive Grid Refinement for the Euler Equations", DOE/ER/03077-202, October 1983.

[22] de Neef T, "Shock Fitting for Everybody", Computers and Fluids, Vol 8, 1980, pp 327-334.

[23] Ecer A and Akay HU, "Investigation of Transonic Flow in a Cascade Using an Adaptive Mesh", AIAA-80-1430, July 1980.

[24] Grossman B and Moretti G, "Time-Dependent Computation of Transonic Flows", AIAA-70-1322, October 1970.

[25] Moretti G, "Experiments in Multi-Dimensional Floating Shock-Fitting", PIBAL Rept 73-18, August 1973.

[26] Moretti G and Abbett M, "A Time-Dependent Computational Method for Blunt Body Flows", AIAAJ, Vol 4, No 12, September 1966, pp 2136-2141.

[27] Carter JE, "A New Boundary-Layer Inviscid Iteration Technique for Separated Flow", AIAA-79-1450, July 1979.

[28] Ni RH, "A Multiple-Grid Scheme for Solving the Euler Equations", AIAAJ, Vol 20, No 11, November 1982, pp 1565-1571.

[29] Peyret R and Taylor TD, Computational Methods for Fluid Flow, Springer-Verlag, New York, 1983, pp 29-31.

[30] Brandt A, "Multilevel Adaptive Computations in Fluid Dynamics", AIAAJ, Vol 18, No 10, October 1980, pp 1165-1172.

[31] Johnson GM, "Convergence Acceleration of Viscous Flow Computations", NASA TM 83039, October 1982.

[32] Chakravarthy SR, "Euler Equations - Implicit Schemes and Implicit Boundary Conditions", AIAA-82-0228, January 1982.

[33] Brandt A, "Multi-Level Adaptive Solutions to Boundary-Value Problems", Math. of Comp., Vol 31, No 138, April 1977, pp 333-390.

[34] Brown JJ, "A Multigrid Mesh-Embedding Technique for Three-Dimensional Transonic Potential Flow Analysis", AIAA-82-0107, January 1982.

[35] Boerstoel JW and Kassies A, "Integrating Multigrid Relaxation into a Robust Fast-Solver for Transonic Potential Flows around Lifting Airfoils", AIAA-83-1885, July 1983.

[36] Duda RO and Hart PE, Pattern Classification and Scene Analysis, Wiley-Interscience, New York, 1973.

[37] Usab WJ, "Embedded Mesh Solutions of the Euler Equation Using a Multiple-Grid Method", Ph.D. Thesis, Massachusetts Institute of Technology, January 1984.

APPENDIX B - Part 2

ADAPTATION OF EQUATIONS AND GRIDS FOR 2-D EULER SYSTEMS

1. Background

The preceding Part 1 presents the essentials of the present adaption concept but with a focus on one-dimensional applications both by way of illustration and as a first test of its utility. In two dimensions it was to be expected that multiple parameters and equations will require modification of the procedure. The essential changes, however, are:

1) The possibility of having directly interacting, effectively overlapping, features, and

2) A reduction in the proportion of the global domain in which dominant features will appear.

The first is the primary difficulty for adaptive equations but perhaps less so for adaptive grids; i.e. equation adaptation will require special clustering algorithms to separate intersecting features. The second suggests that the gains in two dimensions will be greater than those already experienced for one dimension. During the latter part of this year the research has been limited to two-dimensional Euler equations and therefore adaptation to large disturbance regions and discontinuities as features. The intended extension to Navier-Stokes systems effectively will increase the number and kinds of interactions.

The two-dimensional studies have again used the Ni solver for the global algorithm. Density gradient was employed as the fundamental measure of feature presence, and has proven to be adequate in fields with and without discontinuities. A precise level at which adaptation steps are to be taken to improve an evolving field has not been established, but threshold studies have indicated that distributions of field gradients are very good indicators of possible cutoffs that define proper feature scales. Their role and limitations will be apparent in the test solutions to be presented.

2. Threshold Definition of 2-Dimensional Feature

The adaptation procedure is designed to recognize nonuniformities across the domain, for which spatial rate of change provides one measure of scale. Equivalently, local errors often provide similar information for a discrete numerical method. The distribution of relative rates of change, i.e. local values referred to an average over the global domain, has proven to be a reliable indication of feature locations in two dimensions. Each such threshold level corresponds to a certain portion of the domain (fraction of nodes) which exceed that level. Suitable cutoff criteria were explored by study of actual distributions that were obtained from global solutions.

Figures B1(a) and B2(a) show typical distributions that correspond to transonic and supersonic flowfields generated by the Ni method for a circular arc section in a channel. Relatively few of the nodes (approximately 20%) in these pre-adaption cases are associated with above average gradients. The "knee" in such distributions appears close to the point for a slope of -0.2 [indicated by the square symbol], while other controls are the 1.25 threshold level [hexagon] and the 25% of all nodes marker [triangle]. The consequences

of a specific choice of level are shown in the remaining portions of Figures B1 and B2. The 0.50 threshold includes a relatively large number of nodes and makes quite evident the centered and downstream oriented disturbances that are present at transonic and supersonic speeds. The essential point is that higher thresholds rapidly disclose such primary features as shocks and interactions. Imposing limits such as: those nodes for which gradients are at least 1.25, or the flagging of no more than a 0.25 proportion of the total number, together with a knee approximation, together imply a rational decision basis for grid adaptation. The transonic case focuses in on the relatively weak shock at a threshold level between 1.0 and 2.0; in the supersonic case the several discontinuities, reflection, and intersections near the trailing edge are very clearly defined above a threshold of 1.5.

A similar definition of relevant subdomains takes place during any subsequent search for feature locations. The evolving solution, when based on a first level grid adaptation as in Figures B1 and B2, finds sharper discontinuities, results in threshold distributions with clearer knee locations (as in the supersonic example), and suggests a length scale for the next embedded grid.

Lastly, the 0.5 threshold marker [cross] in the figures is included as a measure of that portion of the domain affected very little by disturbances. The influence for a subsonic flow extends over virtually the entire domain. A large upstream, supersonic region is essentially undisturbed (Figure B2) and a characteristic plateau appears.

3. Smoothing

The present solver requires the explicit addition of artificial viscosity, even for cases without any adaptation. The viscosity is necessary globally in order to smooth out those high frequency errors which arise during the interpolation phase of the multiple-grid cycles. A significantly larger amount of smoothing is also required in the vicinity of a shock in order for it to be properly captured.

In the past the high level of smoothing that was required solely near shocks was imposed globally (e.g. Ni, Usab) and resulted in larger smoothing errors than was necessary in regions away from discontinuities. In recognition of this the present research has introduced a globally varying, smoothing coefficient which adjusts the imposed smoothing to the smallest acceptable value.

The new smoothing coefficient distribution is generated by determining an initial coefficient magnitude at each node on the basis of the local density gradient, which is a measure of the shock presence, and then smoothing these coefficients using a Laplacian-type operator in order to ensure stability. For efficient computation the actual implementation involves the integration of a specially devised partial differential equation which makes use of a smoothed value of both the coefficient from the previous iteration and a forcing function based upon the local density gradient of the new iteration.

Use of the spatially varying coefficient requires special care to ensure that the smoothing does not disturb the global conservation properties of the inviscid solver. Conservation is obtained in the current research by using

conservative differencing similar to that discussed by Tong [PhD Thesis, MIT, February 1984]. The embedded mesh interface poses no special smoothing problems as long as post-smoothing is used. Tong points out that for model problems a post-smoothing procedure in fact results in improved convergence rates over those obtainable with traditional co-smoothing. Numerical experiments we have completed corroborate his results for the full two-dimensional Euler equations.

4. Two-Dimensional Shock Fitting

Since we require that the global grid remain fixed, it must be possible for the shock to "float" through the grid with both its location and orientation being arbitrary. This results in significant shock/grid interaction problems.

Software has been written to allow interpolation near shocks with the proper domain of dependence and to track a shock surface as it moves from iteration to iteration. Such movement can result in node crossings as well as simply adjustment of an oblique shock segment within the cell interior. The collapsed feature therefore adds to the pointer burden. In order to accommodate the floating behavior, new elements were added to the pointer system. These store the shock/grid intersections as a "linked list." Effectively node locations have been made aware of adjacent cells to permit the inter-cell transfers. Such transfers may well be few in number after a precise positioning of the shock by several levels of grid adaptation and removal of the fine mesh.

The equation adaptation for collapsed shocks is still being developed and is not included in the adaptive embedded solutions discussed next.

5. Two-Dimensional Adaptive Grid Euler Flow

The threshold study offered encouragement for multiple embedding based on the density variations. Solutions have been completed for subsonic, transonic and supersonic flow with one and two levels of adaptation. They also have been obtained without adaptation for an initially coarse grid, and a globally fine grid which is equivalent in scale to the finest scale with embedding.

The geometry is that of a 10% thick biconvex, circular arc cascade. Figures B3 through B6 show the several transonic solutions in terms of the final grids that were used or adapted, Mach number contours that indicate the increasing gradient that is developed at the shock (which stand at about the three-quarter chord location), and the fractional loss of total pressure contours which are present primarily due to the shock. The coarsest grid does indicate losses at the leading edge as well (Figure B3) and the two levels of adaptation lead to the consistent suggestion of the presence of a relatively larger disturbance there (Figure B5).

The comparison between two-level adaptation and the uniformly fine grid result is reasonably good, but with some differences in the region opposite the airfoil which may result from an excessive threshold choice [see Figures B1(b)-(d)]. The comparable accuracy corresponds to the convergence histories shown in Figures B7 and B8. The abscissas are number of iteration cycles and normalized CPU time respectively, and an appreciable saving (factor of 10) in machine time is apparent when adapting. It is anticipated that the adaptive

equation application will result in additional but somewhat lesser savings.

A supersonic flow (M = 1.4) field for the same geometry (4% thick) is shown in Figure B9 after two levels of adaptation. Reference to Figure B2 suggests that a high threshold was employed. Nevertheless, the Mach number contours make quite clear the intersection of the reflected shock and the shock that is generated at the trailing edge. Without adaptation this very essential behavior is completely overlooked. The sensitivity to threshold level is to be balanced against the increased precision that should result on collapsing the features. That capability should be available shortly.

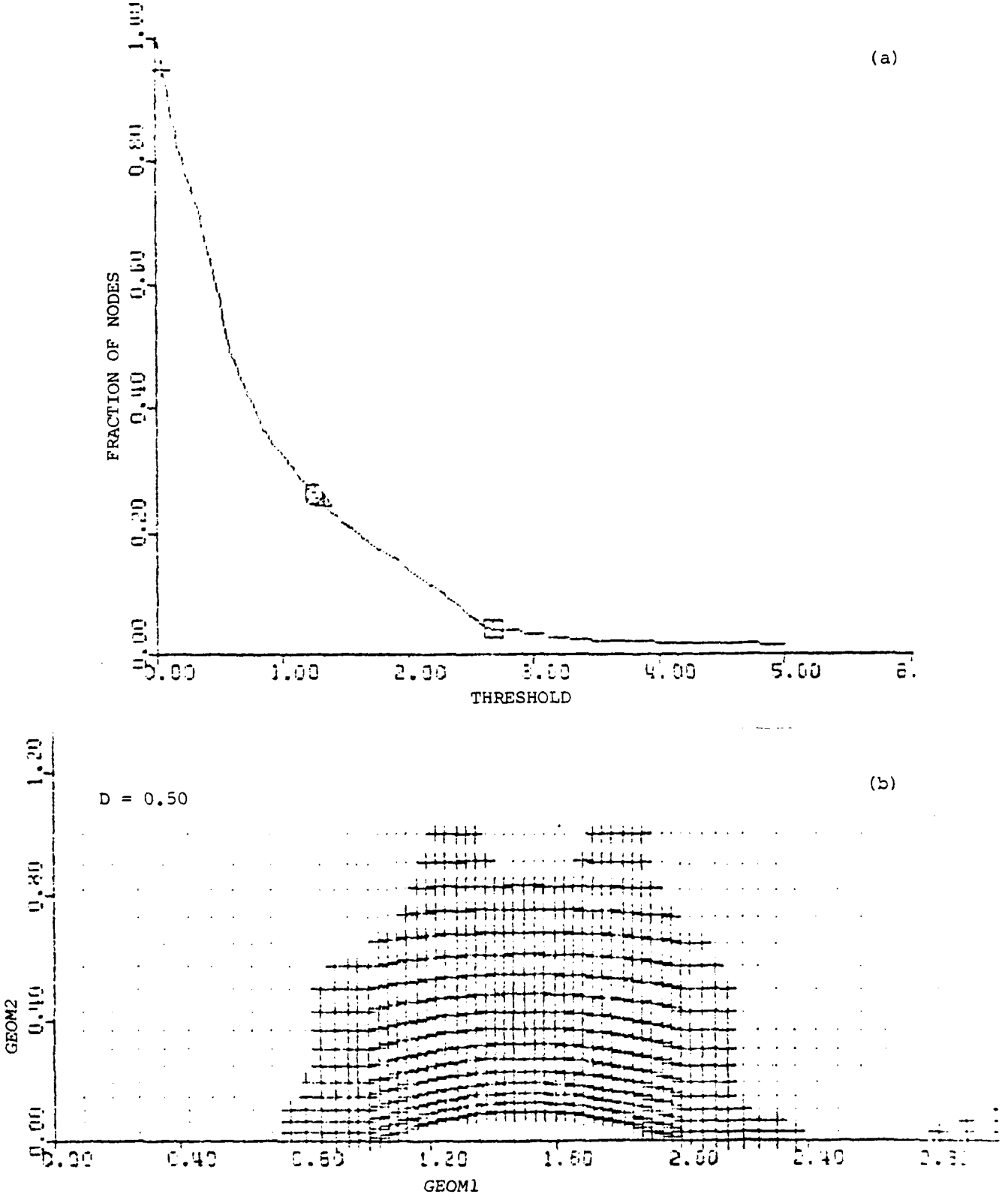
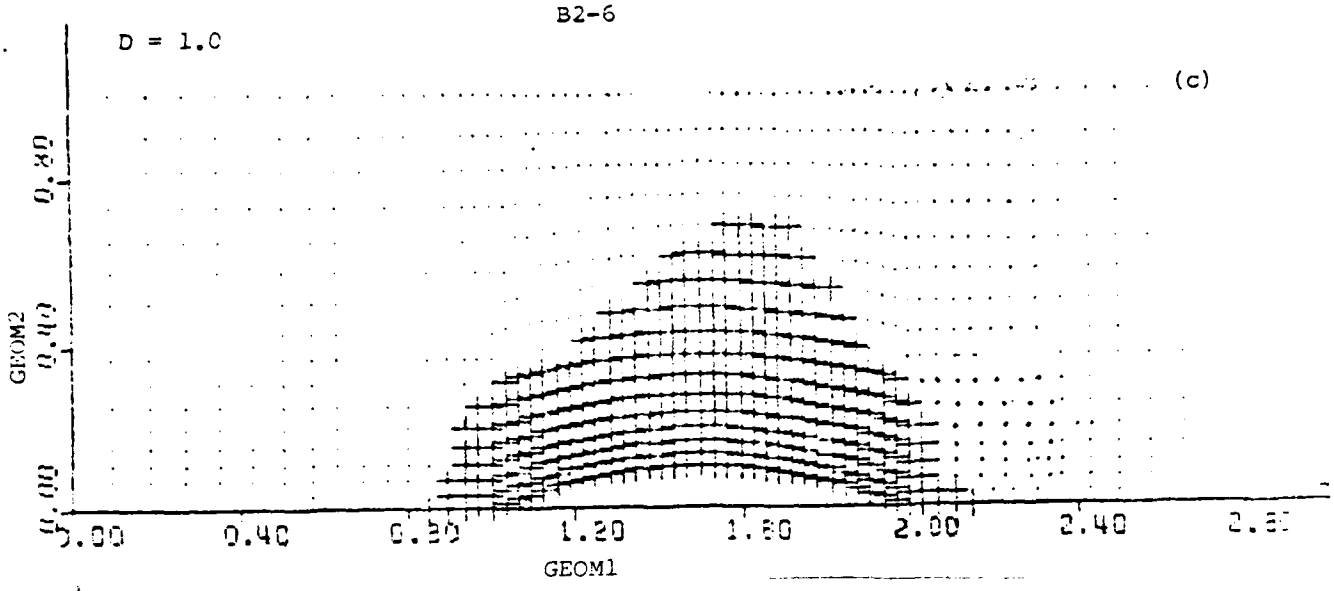
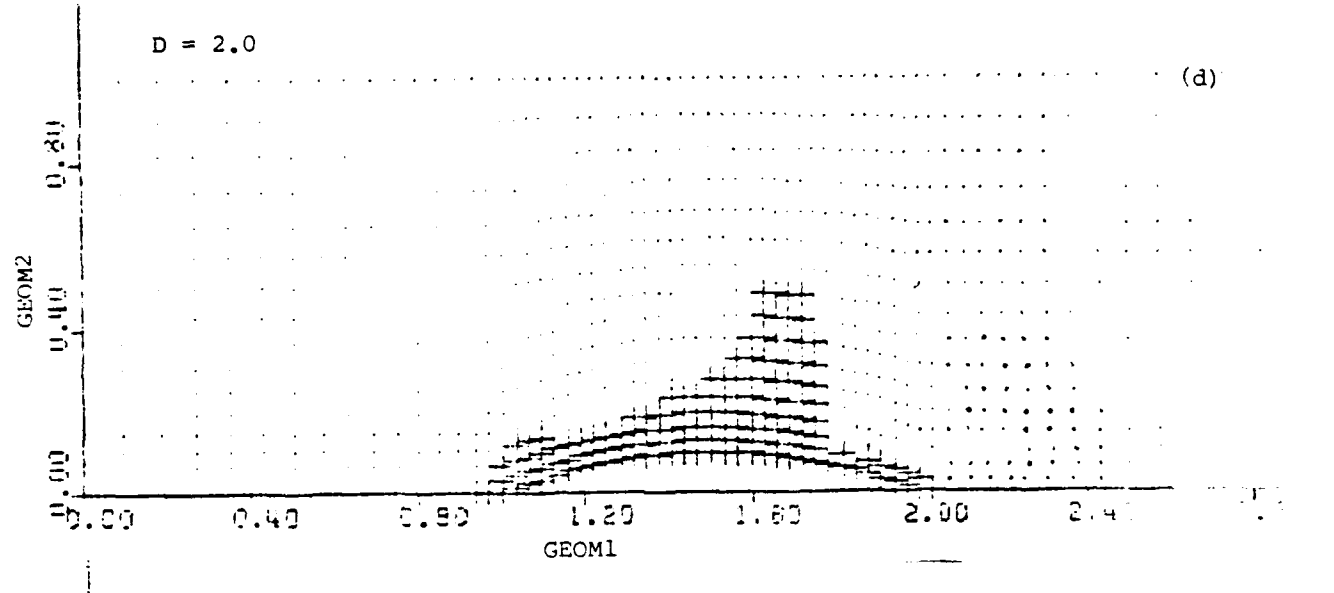


Figure B1. Threshold Influence. Transonic Flow, M = 0.7, 10% bump.
(a) Threshold distribution
(b) Selected level = 0.50





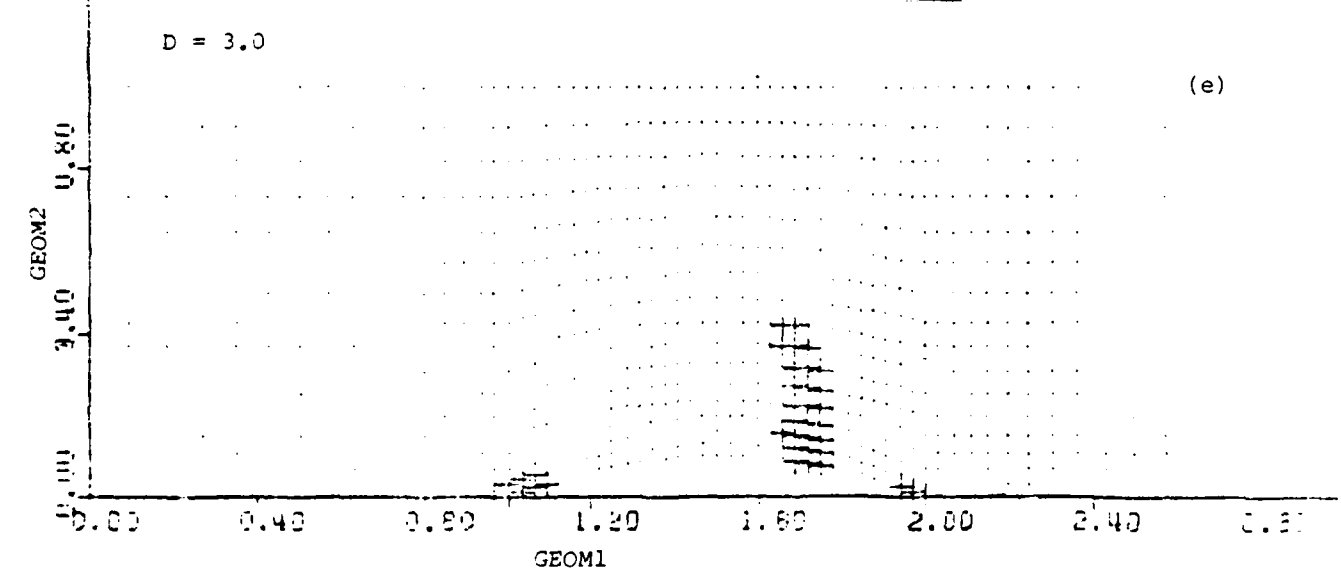
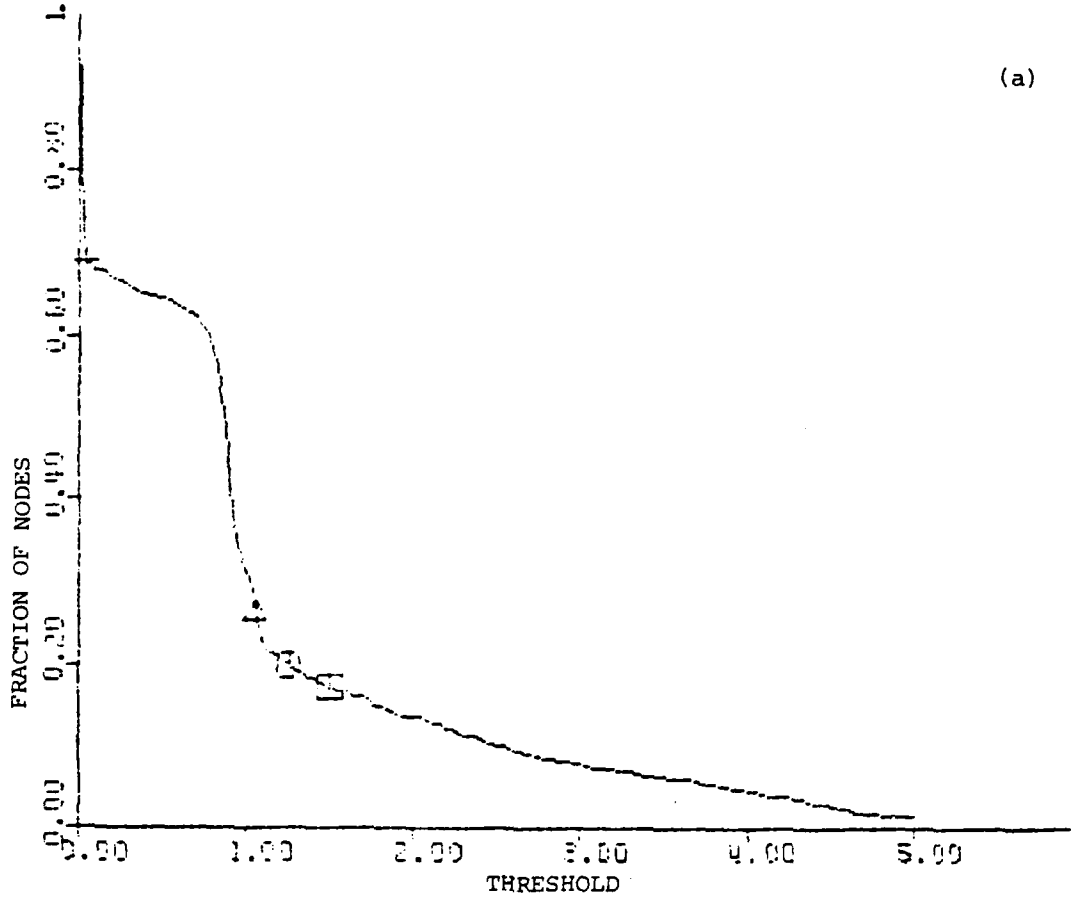


Figure B1 (concluded)

(c) Selected level = 1.0

(d) Selected level = 2.0

(e) Selected level = 3.0



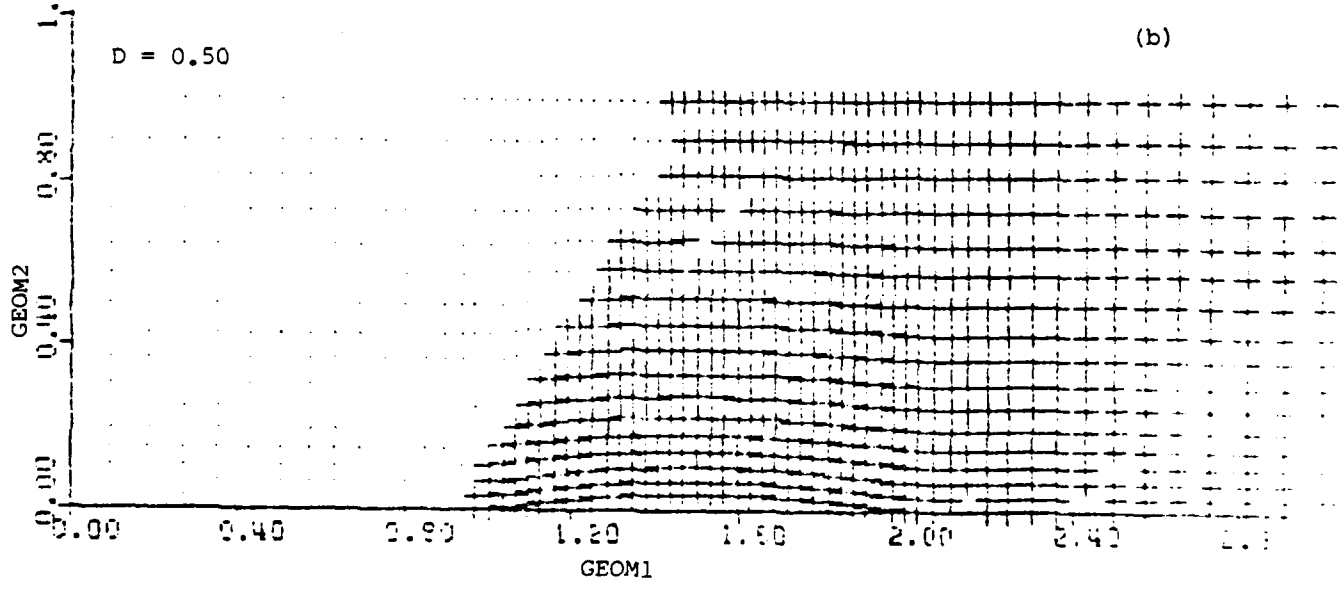
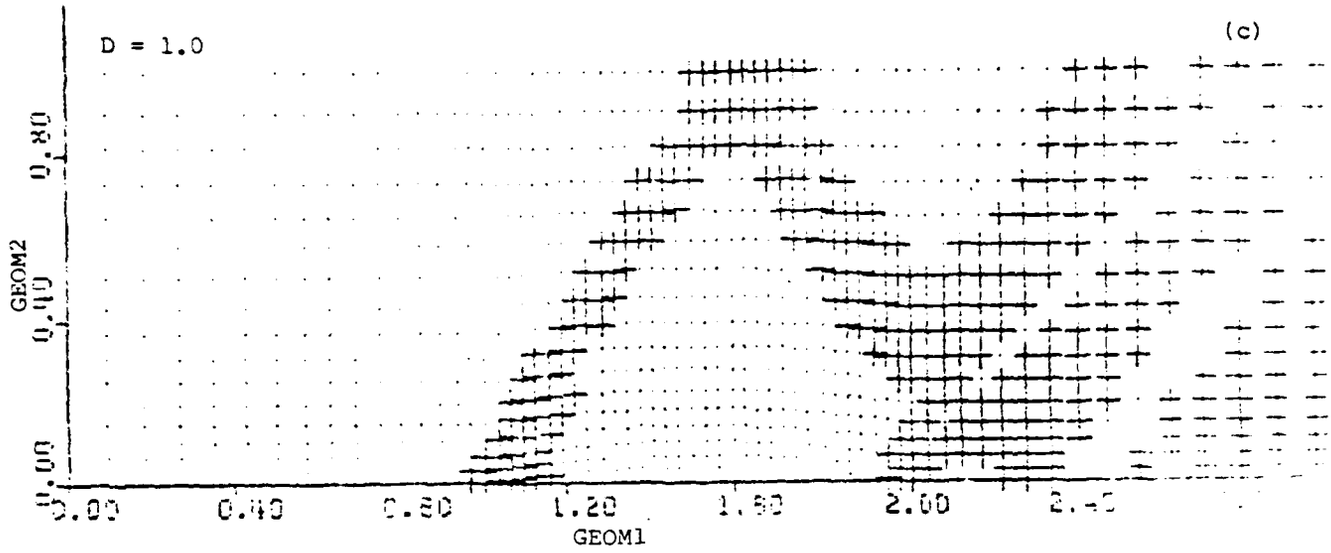
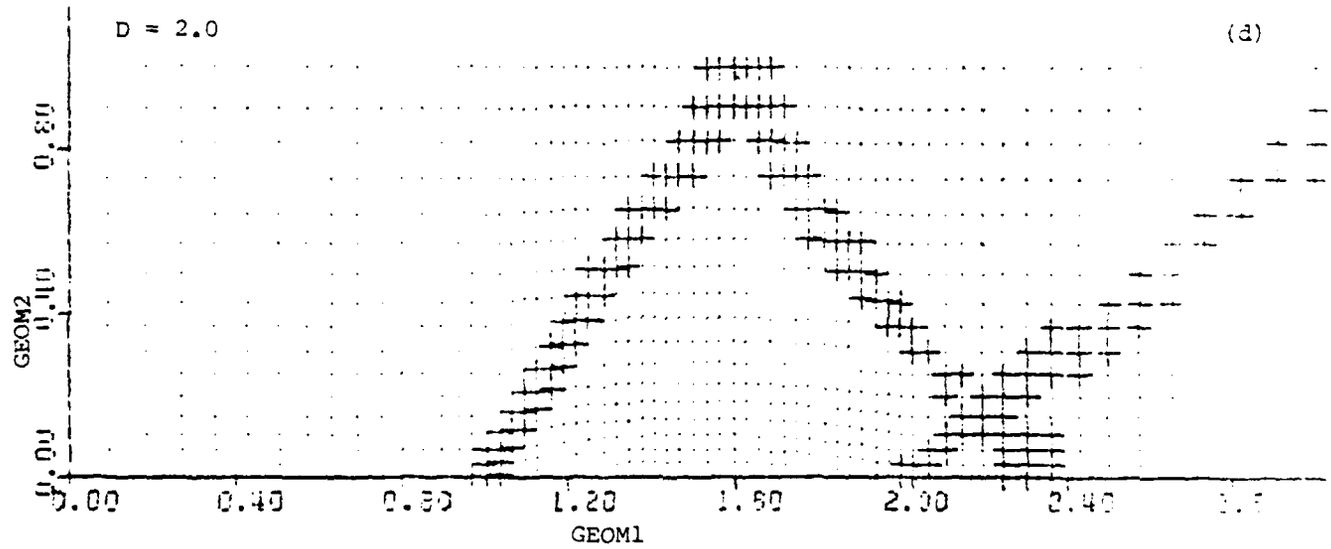


Figure B2. Threshold Influence. Supersonic Flow, M = 1.4, 4% bump
(a) Threshold distribution
(b) Selected level = 0.50





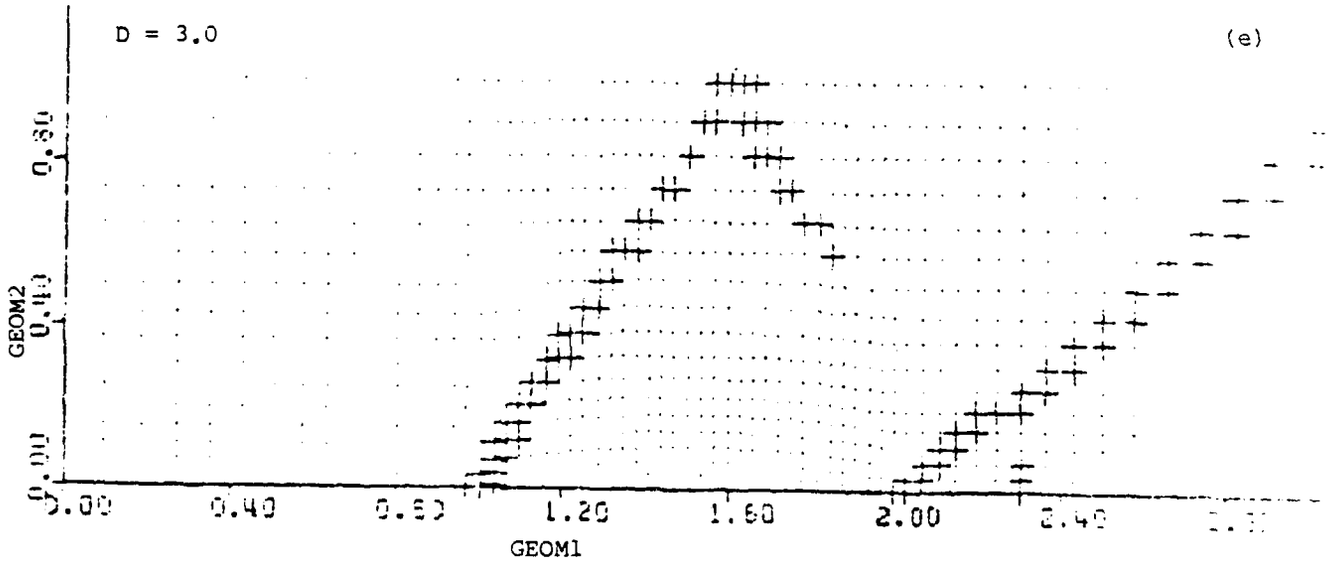
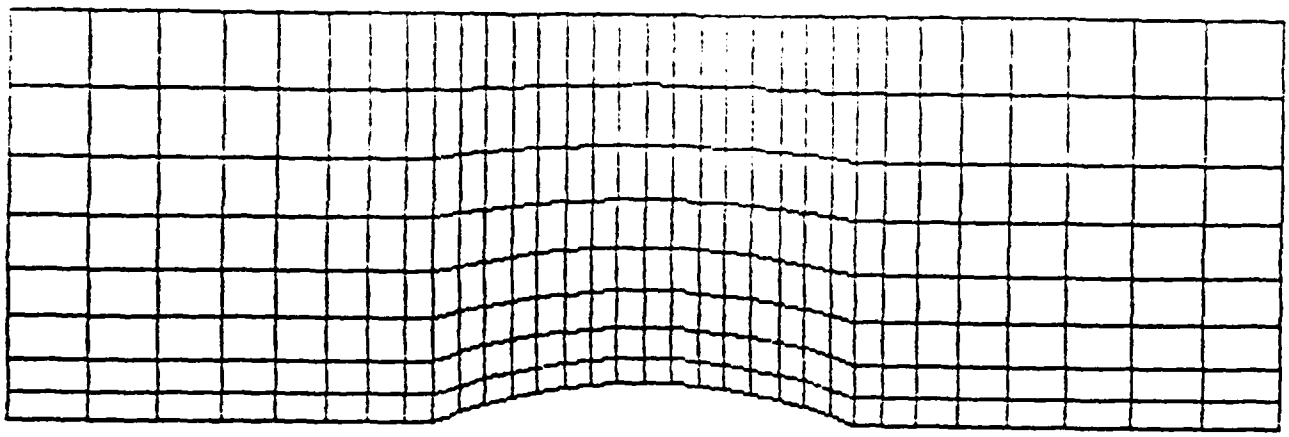
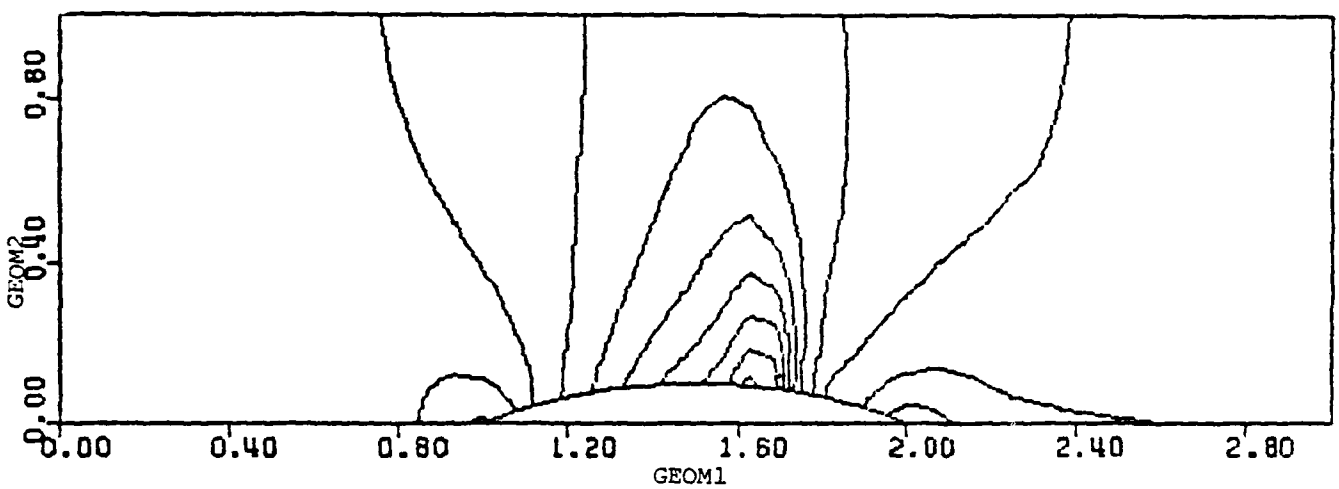


Figure B2 (concluded)

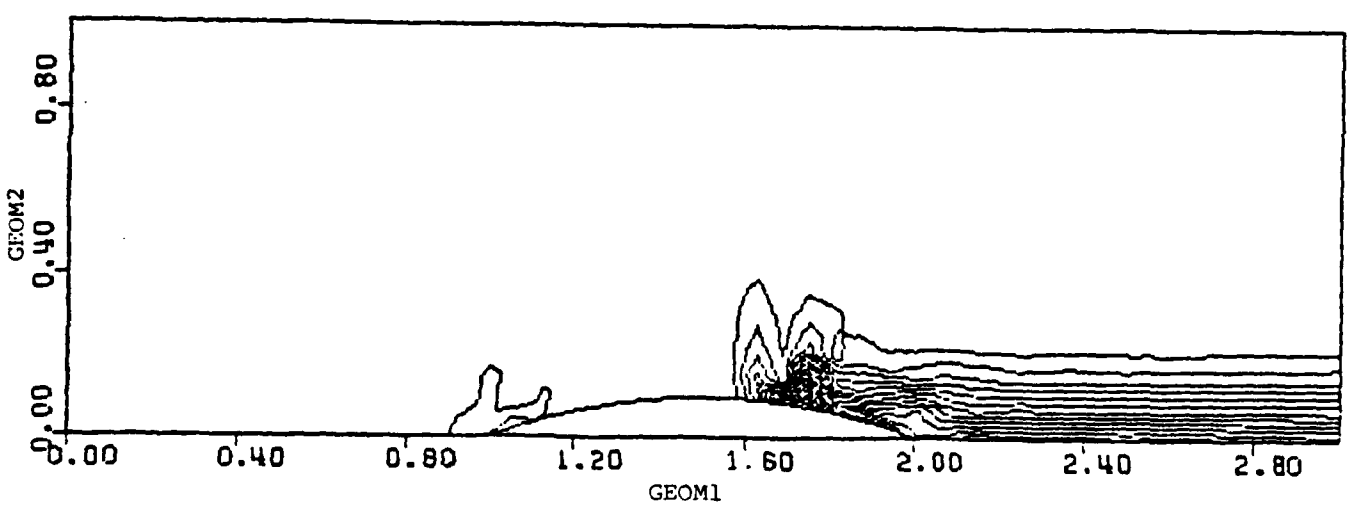
(c) Selected level = 1.0
(d) Selected level = 2.0
(e) Selected level = 3.0



(a)



(b)



(c)

Figure B3. No adaptation. Transonic flow in channel. M = 0.7, 10% bump
(a) Grid
(b) Mach number contours ($\Delta M = 0.1$)
(c) Fractional total pressure loss ($\Delta = 0.005$)

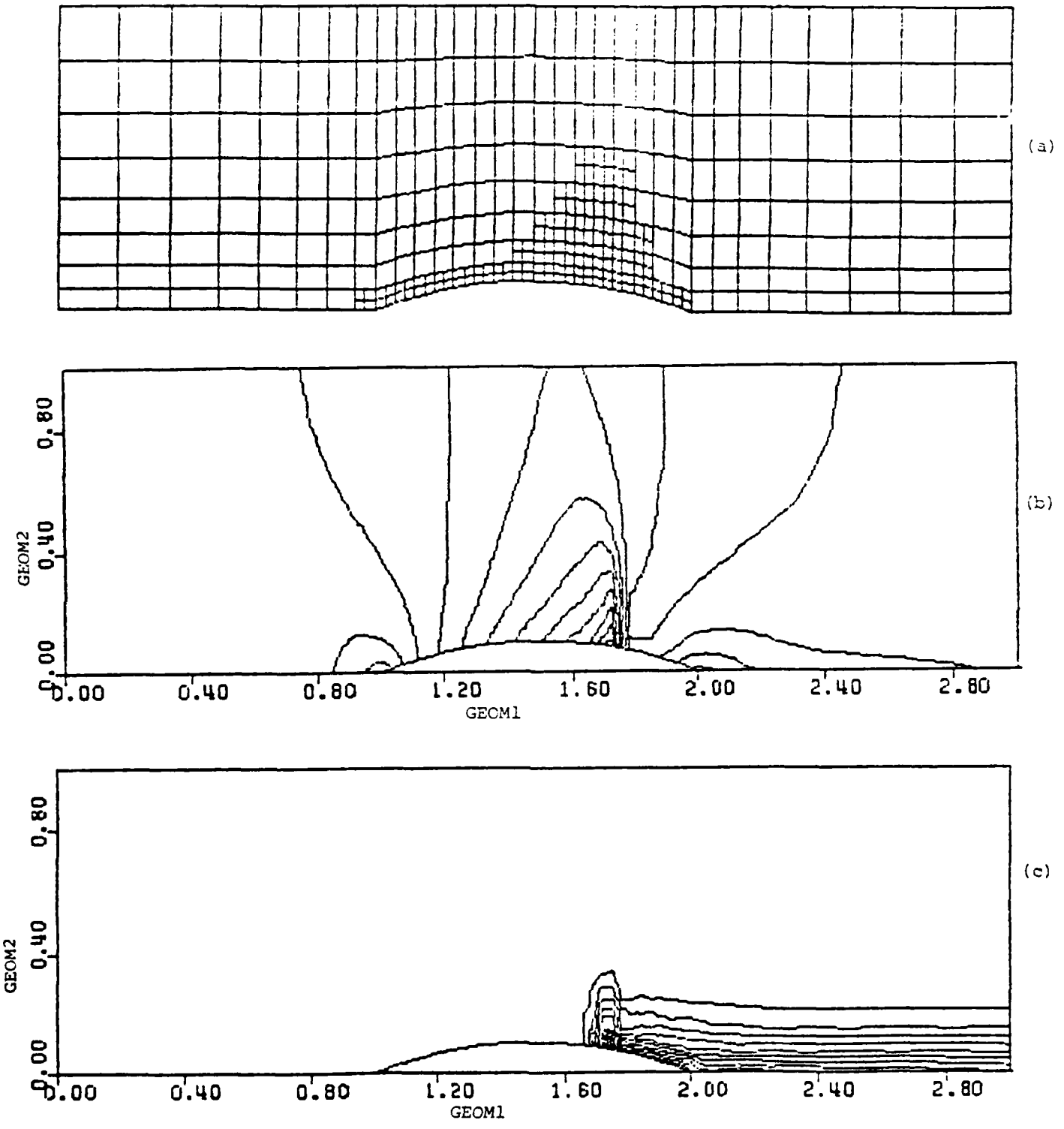
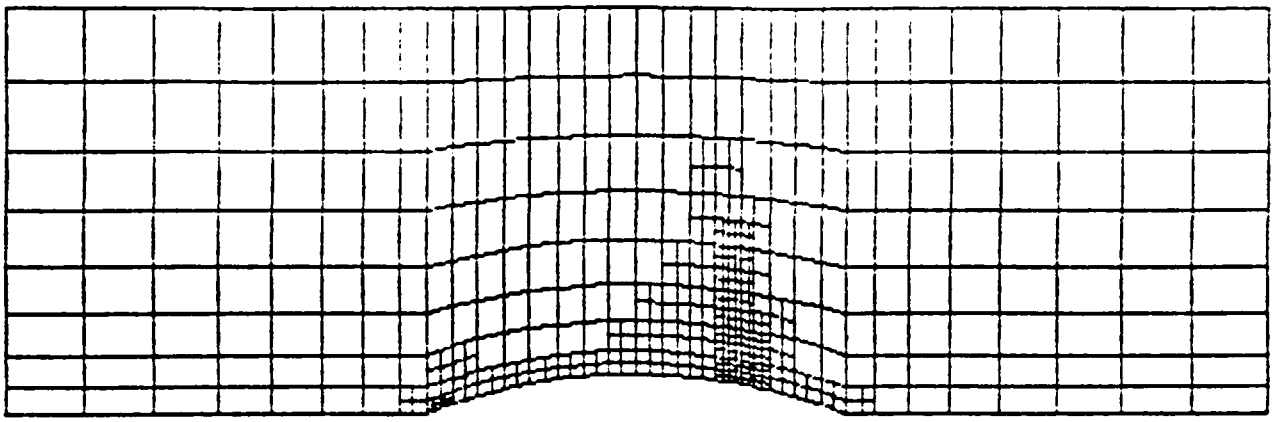
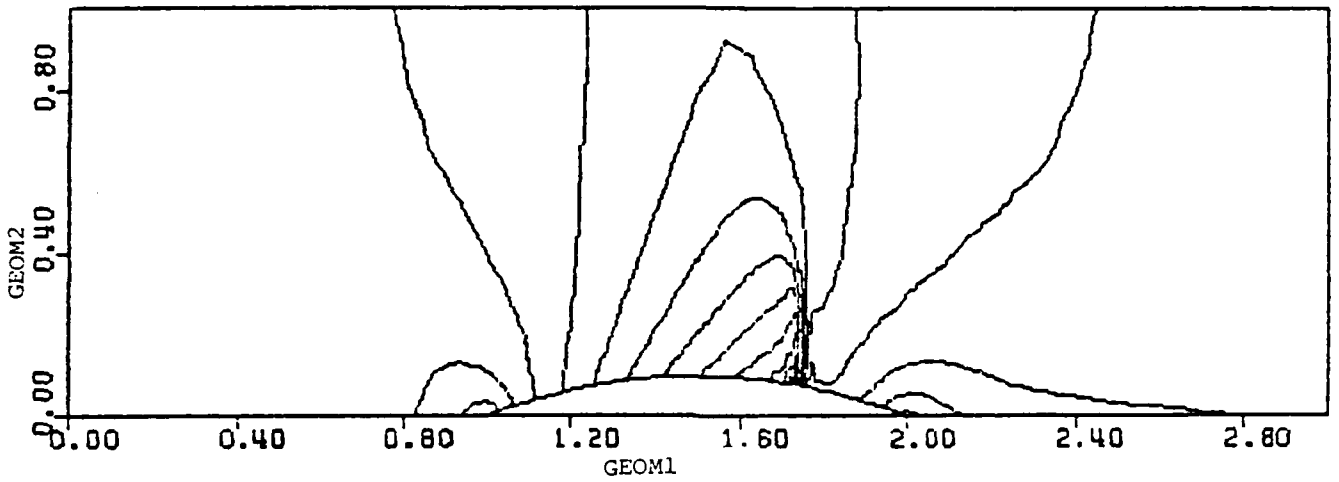


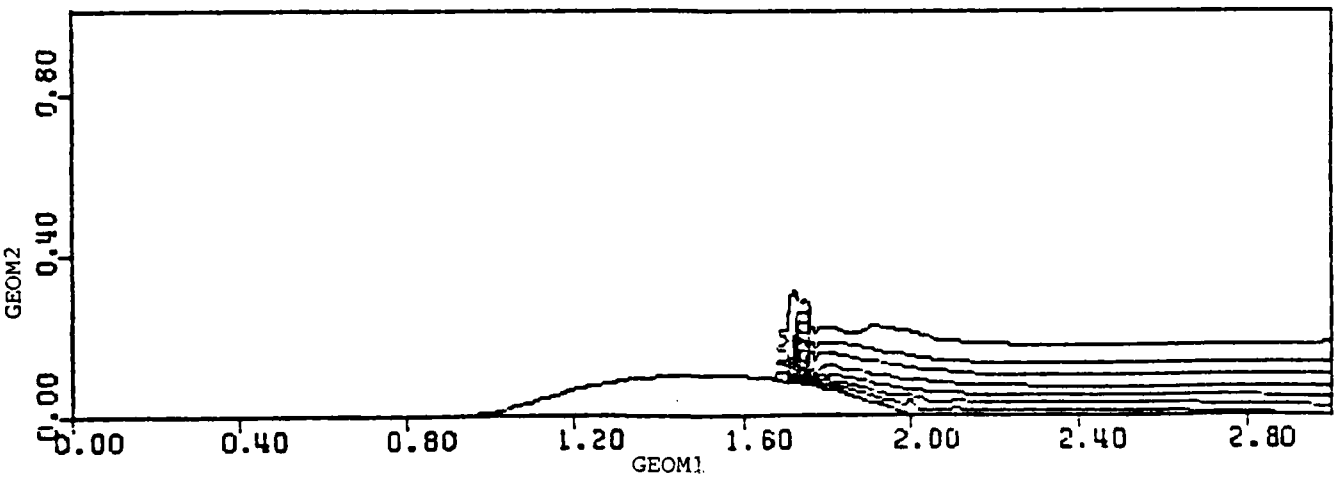
Figure B4. One level of adaptation. Transonic flow in channel. M = 0.7, 10% bump
(a) Grid
(b) Mach number contours ($\Delta M = 0.1$)
(c) Fractional total pressure loss ($\Delta = 0.01$)



(a)

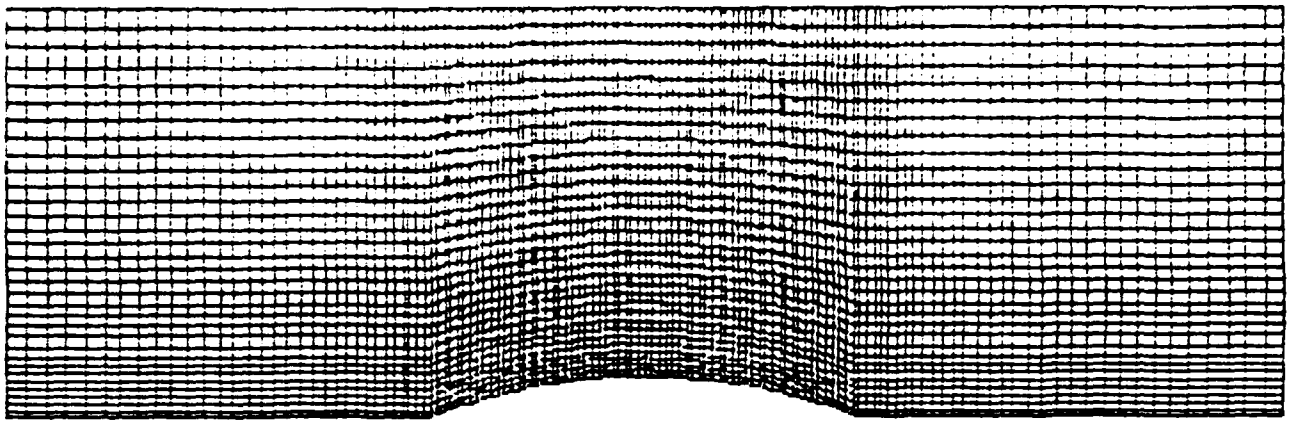


(b)

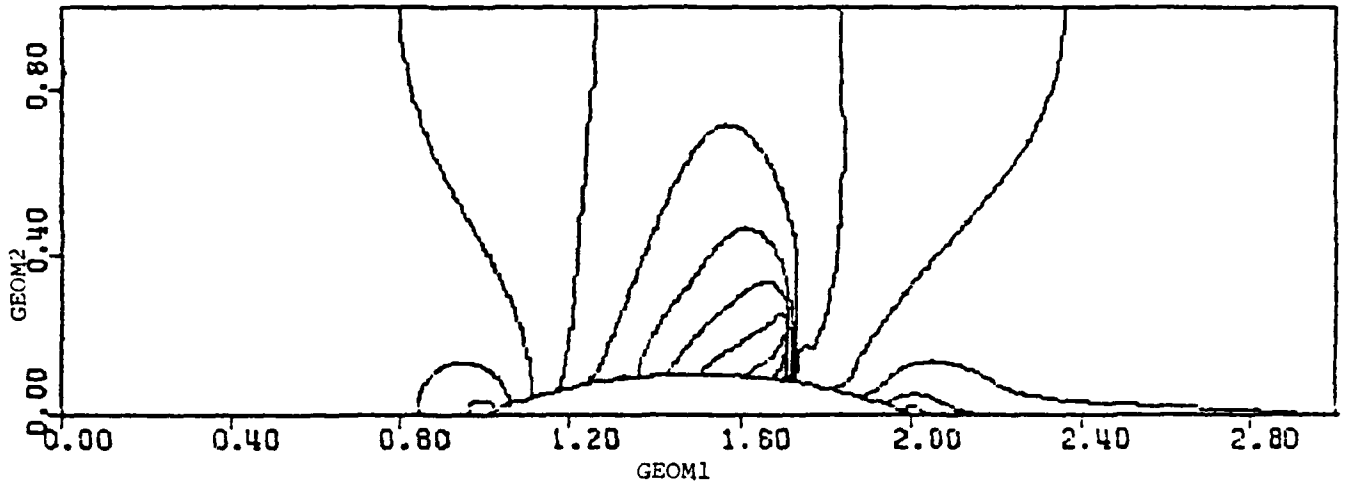


(c)

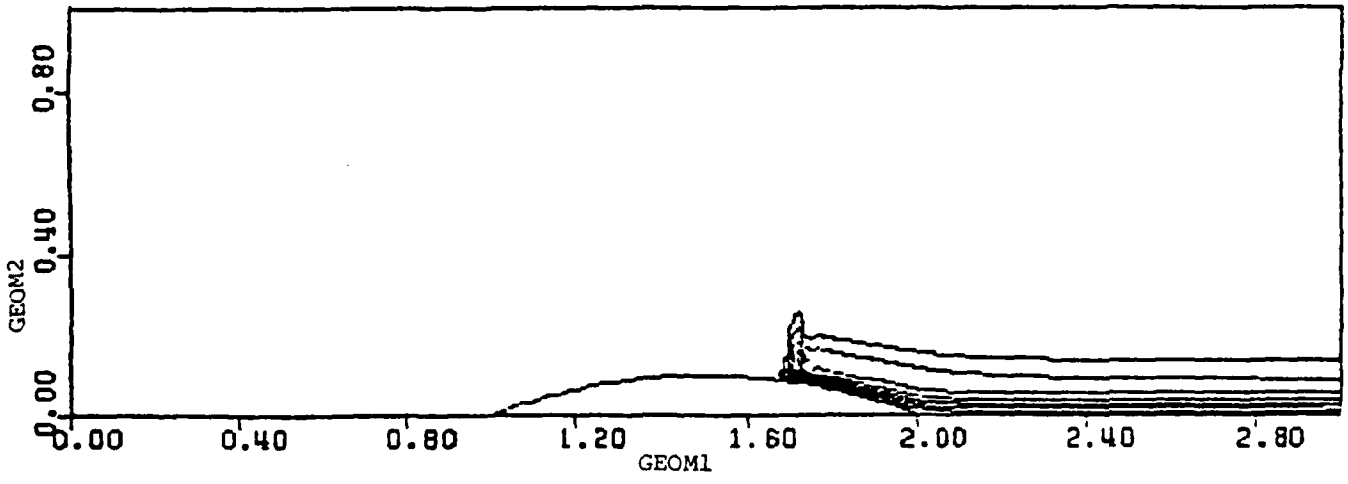
Figure B5. Two levels of adaptation. Transonic flow in channel. M = 0.7, 10% bump
(a) Grid
(b) Mach number contours ($\Delta M = 0.1$)
(c) Fractional total pressure loss ($\Delta = 0.01$)



(a)



(b)



(c)

Figure B6. All cells subdivided twice. Transonic flow in channel. M = 0.7, 10% bump
(a) Grid
(b) Mach number contours ($\Delta M = 0.1$)
(c) Fractional total pressure loss ($\Delta = 0.01$)

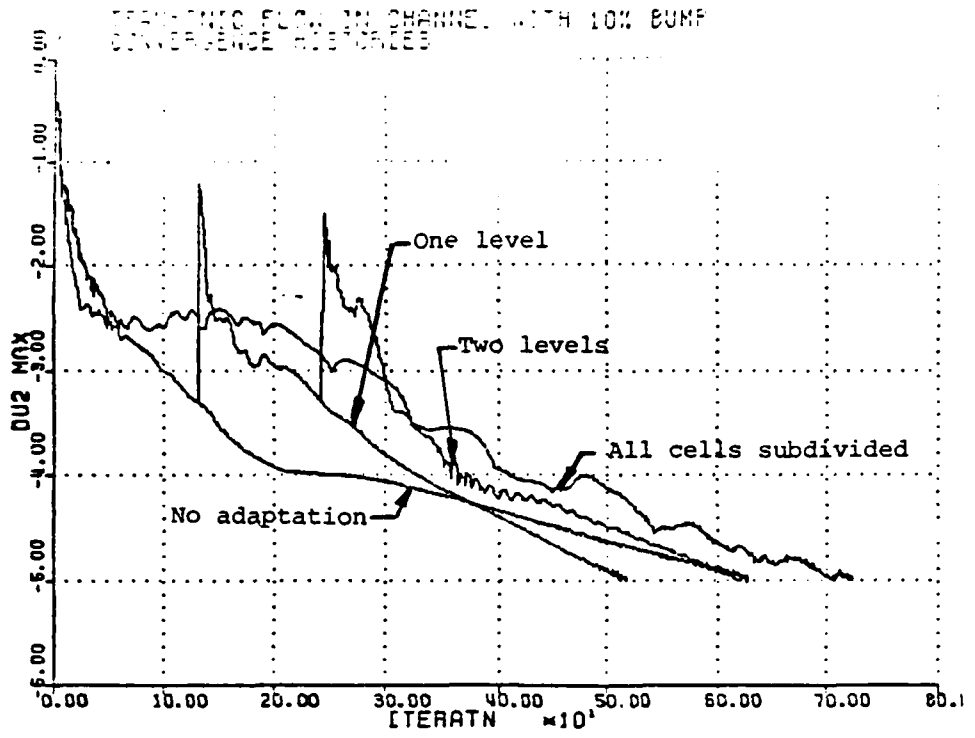


Figure B7. Iteration convergence histories for results in Figures B3-B6

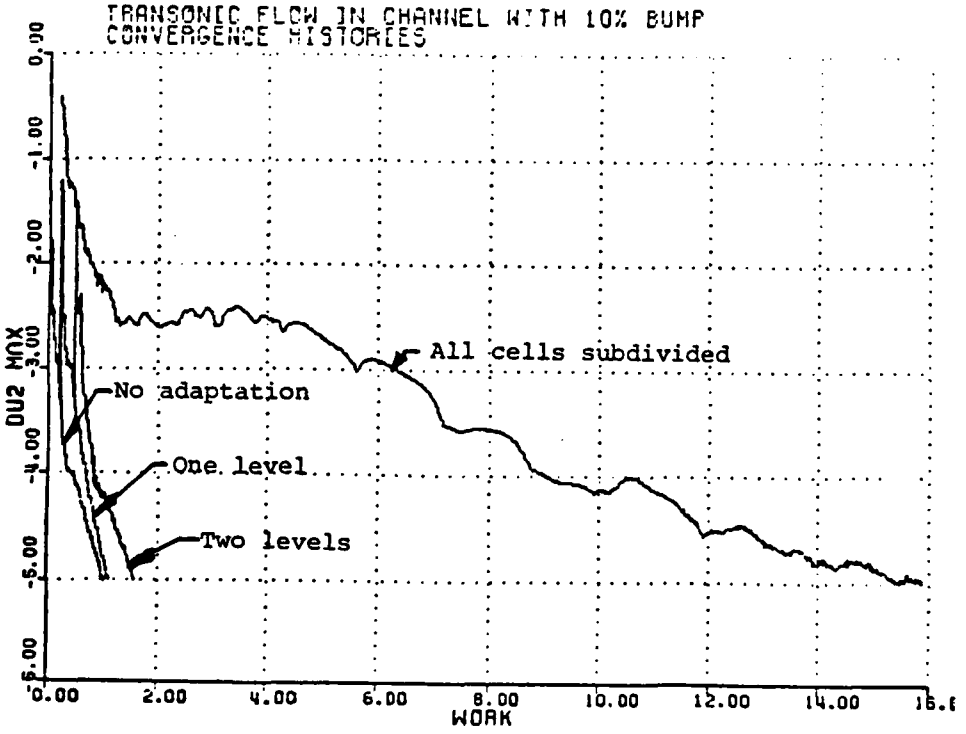
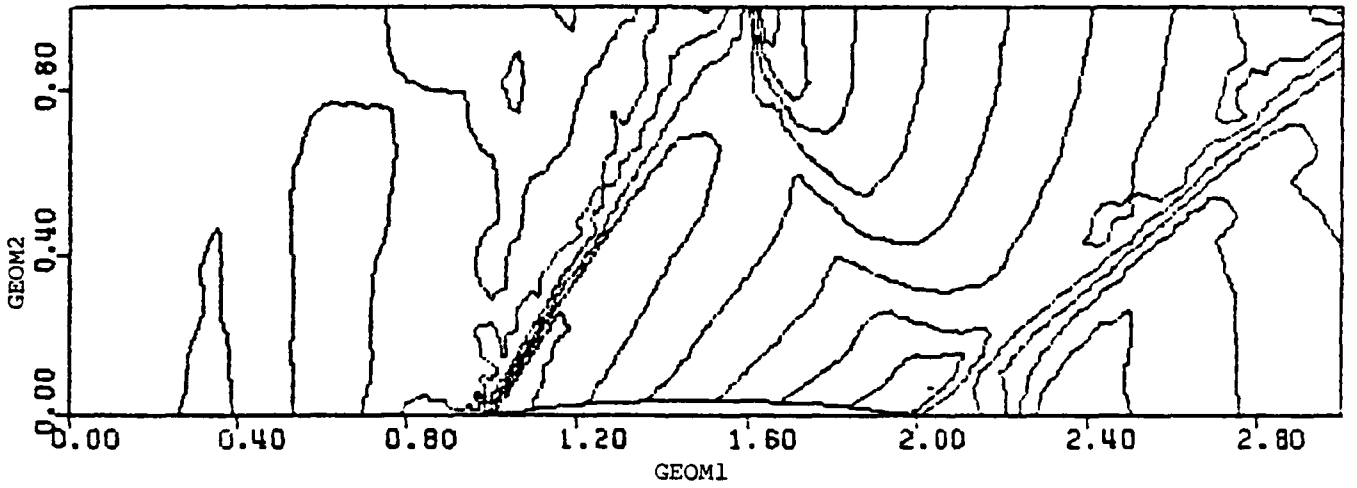
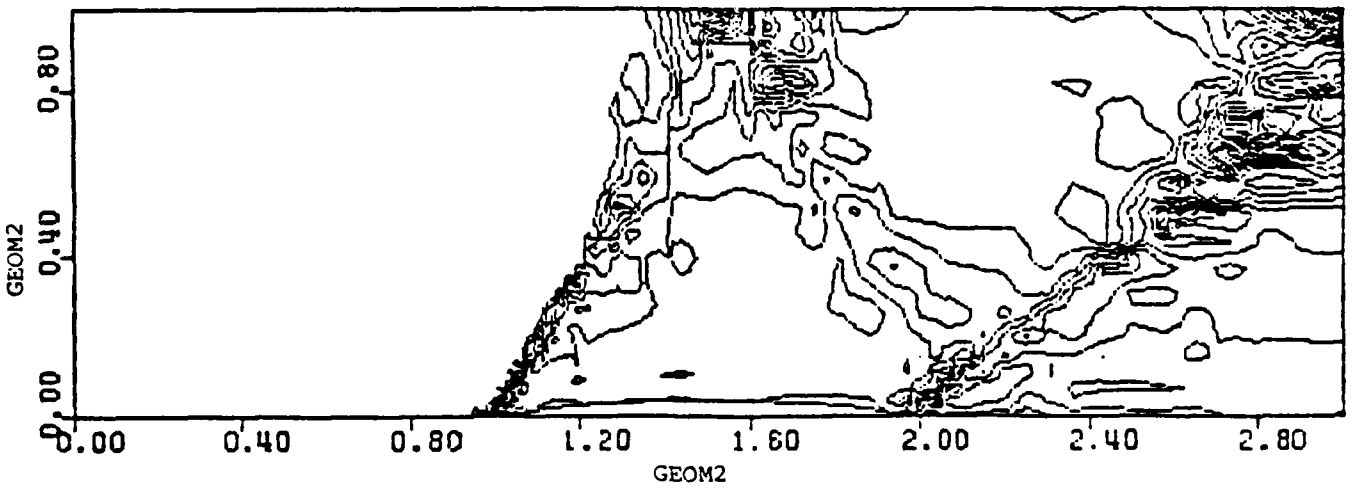


Figure B8. Relative CPU time convergence histories for results in Figures B3-B6.

(a)



(b)



(c)

Figure B9. Two levels of adaptation. Supersonic flow in channel. M = 1.40, 4% bump
(a) Grid
(b) Mach number contours ($\Delta M = 0.1$)
(c) Fractional total pressure loss ($\Delta = 0.005$)

APPENDIX C

ACCELERATING CONVERGENCE TO STEADY STATE

1. INTRODUCTION

In many problems of computational fluid dynamics we are interested in the time independent (steady state) solution. Often, and for a variety of reasons, the steady state solution is approached via the explicitly differenced time dependent equations. This method may be very time consuming; for example, when the grid mesh is highly stretched or if there is present a source term due to chemistry, combustion etc. The reason for the high computer time expenditure is the restricted stability criterion necessary to meet time consistancy. Over the past ten years (at least) some computational fluid dynamicists used a "super-convergence" method which gave up time consistancy in order to impose at each grid point a time step based on the maximal stable Courant number. This approach, though empirical or heuristic at best, yields good results - convergence to steady state is markedly accelerated.

With this background in mind a natural question is - can one determine, rationally and a priori, an optimal distribution of local Courant numbers? The optimum is defined here to mean a distribution of Courant numbers so that at each iteration the steady state residual (or a suitable norm thereof) is diminished by a maximum amount. We next present analyses for the cases of model hyperbolic differential equations. We show that one can do much better than "super convergence".

2. A SCALAR ONE DIMENSIONAL CASE

2.1 The Linear Case

Consider the linear hyperbolic first order partial differential equation with variable coefficients:

$$\frac{\partial u}{\partial t} + c(x)\frac{\partial u}{\partial x} = \sigma(x)u \quad ; \quad 0 \leq x \leq 1 \ , \quad t \geq 0 \tag{1}$$

$$u(0,t) = u_0 \ .$$

For the moment assume $c(x) > 0$, $\sigma(x) \geq 0$. The finite difference approximation will be taken to be a one-sided (up-wind) first order differencing in x and a simple single level time differencing. The grid is *not* necessarily uniform and since we are giving up time consistency the time step at each grid point j, at each iteration count n, Δt_j^n, will not be a constant over the j's. The finite difference representation of Eq. (1) is, then

$$u_j^{n+1} = u_j^n - \frac{\Delta t_j^n c_j}{\Delta x_j}(u_j^n - u_{j-1}^n) + \Delta t_j^n \sigma_j u_j^n \ . \tag{2}$$

Define local Courant and source numbers respectively:

$$\lambda_j^n = \frac{\Delta t_j^n c_j}{\Delta x_j} \tag{3}$$

$$\alpha_j = \frac{\sigma_j \Delta x_j}{c_j} \ . \tag{4}$$

Eq. (1) becomes:

$$u_j^{n+1} = u_j^n - \lambda_j^n(u_j^n - u_{j-1}^n) + \lambda_j^n \alpha_j u_j^n \tag{5}$$

or, more compactly

$$u_j^{n+1} = u_j^n - \lambda_j^n L u_j^n \tag{6}$$

the definition of the differencing operation L being clear from Eqs. (5) and (6).

By steady state we mean that $Lu_j^n = 0$, or equivalently $u_j^{n+1} = u_j^n$. Consider an iteration level n: usually $Lu_j^n \neq 0$. We will, however, be satisfied if the L_2 - norm will be less than some specified level; i.e. $\|Lu_j^n\|^2 < \varepsilon$. We assume that this criterion for steady state is not met after n iterations (if it is, then the computation is finished). We then ask the question - what is the distribution of the local λ_j^n's such that the norm of the residual at the *next* iteration, $\|Lu_j^{n+1}\|^2$, is minimized. This is a standard least square fit problem, but one has to watch for the boundary condition $u_0^n = u_0$. We write

$$Q^{n+1} \equiv \|Lu_j^{n+1}\|_{L_2}^2 = \|L(u_j^n - \lambda_j^n Lu_j^n)\|_{L_2}^2 = \frac{1}{N}\sum_{j=1}^{N}\{L(u_j^n - \lambda_j^n Lu_j^n)\}^2$$

and substituting from (6) and (5) we have

$$\begin{aligned} Q^{n+1} = {} & \frac{1}{N}\{(1-\alpha_1)[u_1^n - \lambda_1^n(u_1^n - u_0) + \lambda_1^n\alpha_1 u_1^n] - u_0\}^2 \\ & + \frac{1}{N}\sum_{j=2}^{N}\{(1-\alpha_j)[u_j^n - \lambda_j^n(u_j^n - u_{j-1}^n)] + \lambda_j^n\alpha_j u_j^n] \\ & \qquad - [u_{j-1}^n - \lambda_{j-1}^n(u_{j-1}^n - u_{j-2}^n) + \lambda_{j-1}^n\alpha_{j-1}u_{j-1}^n]\}^2 . \end{aligned} \tag{7}$$

Differentiating Q^{n+1} with respect to λ_j^n $(1 \leq j \leq N)$ leads to a set of N linear algebraic equations:

$$-(1-\alpha_1)^2[u_1^n - \lambda_1^n(u_1^n - u_0^n - \alpha_1 u_1^n)] + (1-\alpha_1)u_0 + z_2 = 0 \tag{8}$$

$$z_{j+1}^n - (1-\alpha_j)z_j^n = 0 \qquad (2 \leq j \leq N-1) \tag{9}$$

$$z_N^n = 0 \quad , \tag{10}$$

where

$$z_j^n = (1-\alpha_j)[u_j^n - \lambda_j^n(u_j^n - u_{j-1}^n) + \lambda_j^n \alpha_j u_j^n]$$

$$- [u_{j-1}^n - \lambda_{j-1}^n(u_{j-1}^n - u_{j-2}^n) + \lambda_{j-1}^n \alpha_{j-1} u_{j-1}^n] \quad . \tag{11}$$

It is seen immediately that

$$z_j^n = 0 \qquad (2 \leq j \leq N) \quad . \tag{12}$$

We can then compute λ_1^n from (8) and the rest of the λ_j's recursively from (11). Specifically, we find

$$\lambda_1^n = 1/\omega_1 \quad . \tag{13}$$

and

$$\lambda_j^n = \frac{(\prod_{s=1}^{j} \omega_s) u_j^n - u_0}{(\prod_{s=1}^{j} \omega_s)(u_j^n - u_{j-1}^n - \alpha_j u_j^n)} \qquad (2 \leq j \leq N) \tag{14}$$

where

$$\omega_j = (1-\alpha_j) \quad .$$

If we substitute λ_j^n from (14) into the governing difference Eq. (5) we find

$$u_j^{n+1} = u_0 / \prod_{s=1}^{j} \omega_s \quad . \tag{15}$$

The R.H.S. of (15) is the *steady state solution* of (5) and thus we find that the least square minimization approach yields the steady state in one single iteration.

2.2 A Non-linear Example

If, instead of (1), we have

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = \sigma(x)u \tag{16}$$

with $f = \frac{1}{2}u^2$, the difference equation, by analogy to (2) is

$$u_j^{n+1} = u_j^n - \lambda_j^n(\tfrac{1}{2}u_j^{n^2} - \tfrac{1}{2}u_{j-1}^{n^2}) + \lambda_j^n\alpha_j u_j^n \quad . \tag{17}$$

The least square minimization in this case leads to:

$$\lambda_1^n = \frac{u_1 - \alpha_1 - \sqrt{u_0^2+\alpha_1^2}}{\frac{1}{2}u_1^2 - \frac{1}{2}u_0^2 - \alpha_1 u_1} \tag{18}$$

and for $j \geq 2$ we have recursively

$$\lambda_j^n = \frac{u_j^n - w_j}{\frac{1}{2}u_j^{n^2} - \frac{1}{2}u_{j-1}^{n^2} - \alpha_j u_j} \tag{19}$$

where

$$w_1 = \alpha_1 + \sqrt{\alpha_1^2+u_0^2}$$

$$w_j = \alpha_j + \sqrt{\alpha_j^2+w_{j-1}^2} \qquad (2 \leq j \leq N) \quad .$$

In this case also it can be demonstrated that with this choice of λ_j^n's steady state is achieved after one iteration (n=1).

3. EXAMPLE OF A SYSTEM

Consider the 2x2 linear system (obtained after diagonalizing a more general system)

$$\begin{aligned} u_t + c(x)u_x &= a(x)u + b(x)v \\ v_t + e(x)v_x &= A(x)u + B(x)v \end{aligned} \qquad c(x),\ e(x) > 0 \tag{20}$$

The upwind difference equations are:

$$\begin{aligned} u_j^{n+1} &= u_j^n - \lambda_j^n[(u_j^n - u_{j-1}^n) - \alpha_j u_j^n - \beta_j v_j^n] = u_j^n - \lambda_j^n L_j^n(u,v) \\ v_j^{n+1} &= v_j^n - \mu_j^n[(v_j^n - v_{j-1}^n) - \gamma_j v_j^n - \delta_j u_j^n] = v_j^n - \mu_j^n M_j^n(u,v) \end{aligned} \tag{21}$$

where

$$\lambda_j^n = \frac{\Delta t_j^n c_j}{\Delta x_j} \quad ; \quad \mu_j^n = \frac{\Delta \tau_j^n e_j}{\Delta x_j} \quad ; \quad \alpha_j = \frac{a_j \Delta x_j}{c_j}$$

$$\beta_j = \frac{b_j \Delta x_j}{c_j} \quad ; \quad \gamma_j = \frac{A_j \Delta x_j}{e_j} \qquad \delta_j = \frac{B_j \Delta x_j}{e_j}$$

$$L_j^n(u,v) = (1-\alpha_j)u_j^n - u_{j-1}^n - \beta_j v_j^n$$

$$M_j^n(u,v) = (1-\gamma_j)v_j^n - v_{j-1}^n - \delta_j u_j^n \quad .$$

Notice that not only we advance u_j^n with different Δt_j^n but the v_j^n is advanced, even at the same j, with a different "time-step" $\Delta \tau_j^n$.

The analysis in this case is more elaborate but basically we least-square minimize the norms of L_j^{n+1} and M_j^{n+1} . The results of this analysis lead to the following expressions for the optimal "Courant numbers" and hence "time-steps":

$$\lambda_1^n = \frac{[(1-\alpha_1)(1-\gamma_1)-\beta_1\delta_1]u_1^n-(1-\gamma_1)u_0-\beta_1 v_0}{[(1-\alpha_1)u_1^n-u_0-\beta_1 v_1^n][(1-\alpha_1)(1-\gamma_1)-\beta_1\delta_1]}$$

$$\mu_1^n = \frac{[(1-\alpha_1)(1-\gamma_1)-\beta_1\delta_1]v_1^n-(1-\alpha_1)v_0-\delta_1 u_0}{[(1-\gamma_1)v_1^n-v_0-\delta_1 u_1^n][(1-\alpha_1)(1-\gamma_1)-\beta_1\delta_1]} \qquad (22)$$

λ_j^n and μ_j^n are found recursively as follows:

$$\lambda_j^n = \frac{[(1-\alpha_j)(1-\gamma_j)-\beta_j\delta_j]u_j^n-(1-\gamma_j)[u_{j-1}^n-\lambda_{j-1}^n L_{j-1}^n]-\beta_j[v_{j-1}^n-u_{j-1}^n M_{j-1}^n]}{[(1-\alpha_j)u_j^n-u_{j-1}^n-\beta_j v_j^n][(1-\alpha_j)(1-\gamma_j)-\beta_j\delta_j]} \qquad (23)$$

and

$$\mu_j^n = \frac{[(1-\alpha_j)(1-\gamma_j)-\beta_j\delta_j]v_j^n-(1-\alpha_j)[v_{j-1}^n-\mu_{j-1}^n M_{j-1}^n]-\delta_j[u_{j-1}^n-\lambda_{j-1}^n L_{j-1}^n]}{[(1-\gamma_j)(v_j^n-v_{j-1}^n-\delta_j u_j^n][(1-\alpha_j)(1-\gamma_j)-\beta_j\delta_j]} \qquad (24)$$

4. NUMERICAL EXPERIMENTS

The question that faced us was: will the λ_j^n's, predicted by the linear theory, be effective in reducing the number of iterations in the case of a non-linear problem. The non-linear theory basically requires the same amount of the work to determine the λ_j^n's as to solve the *steady state* non-linear p.d.e.; hence, the motivation to try the efficacy of the linear theory when applied to the non-linear case. The case tested was that described by Eq. (17) in the range $0 \leq x \leq 1$. The source term was chosen to be:

$$\alpha_j = \frac{1}{\tau} e^{-a(x_j - \frac{1}{2})^2} \tag{25}$$

and the mesh was stretched by the transformation

$$x_j = \tfrac{1}{2}[1 - \frac{1}{\xi}(\nu - \sqrt{\nu^2 + \xi^2})] \tag{26}$$

where

$$\xi = \tfrac{1}{2} \ell n(j/N-j) \tag{27}$$

and ν is the grid stretching parameter and N is the number of grid intervals. The grid stretching is quite sensitive to ν. For example, for $\nu = 2, 5, 10$ and $N = 50$ the following geometry emerges:

ν	$\Delta x_{max} = \Delta x_1 = \Delta x_{50}$	$\Delta x_{min} = \Delta x_{25} = \Delta x_{26}$	$\Delta x_{max}/\Delta x_{min}$
2	.2969	.0050	59.4
5	.40613	.00200	203.1
10	.45180	.00100	451.8

The test problem was run with the parameters: $\tau = 1$, $a = 4$ and $\nu = 2, 5, 10$. Each of the cases was run twice - once with the local Courant number taken to be unity ($\lambda_j^n = 1$) and once with λ_j^n chosen according to the linear theory, Eqs. (13) and (14) ($\lambda_j^n = \lambda_{j,theory}^n$), although the problem has a non-linear advection term. The iterations continue until the L_2-norm of the steady state residual decreased below 10^{-5}. The results are summarized in the following table:

	$\lambda_j^n = 1$	$\lambda_j^n = \lambda_{j,\text{theory}}^n$
ν	iteration count, n.	iteration count, n
2	54	4
5	55	5
10	55	6

5. SUMMARY

It is seen that the number of iterations necessary to converge to steady state of the non-linear model problem with a highly stretched grid and a source term may be reduced by an order of magnitude compared to the "superconvergence" method of taking $\lambda_j^n = 1$, by selecting the local time steps appropriately. Further work is necessary to establish the limits of applicability of the linear prediction to non-linear problems. Also the efficacy of the method in the case of a system or multidimensional problem has to be explored, even though the extension of the linear theory of those cases is straightforward if somewhat elaborate. The theory should also be extended to the case of central finite differences with boundary conditions specified according to the "inflow" and "outflow" conditions. This program poses a realistic goal to the linear theory - its applicability to "real-life" computations will need careful exploration.